Homework4

Sherine George

27 November, 2023

The data in diabetes.csv - also hosted at https://www.kaggle.com/datasets/ uciml/pima-indians-diabetes-database - contains information about female patients of Pima Indian heritage who are at least 21 years old. The data contains the following variables: • Pregnancies: number of pregnancies experienced by the patient • Glucose: the plasma glucose concentration measured from an oral glucose tolerance test (in mg/dL) • BloodPressure: the patient's diastolic blood pressure (in mmHg) • SkinThickness: the skin fold thickness of the patient's triceps (in mm) • Insulin: the patient's serum insulin level (in U/ml) • BMI: the patient's Body Mass Index (in kg/m2) • DiabetesPedigreeFunction: a measure of the likelihood that the patient will develop diabetes based on family history • Age: the patient's age (in completed years) • Outcome: whether or not the patient was diagnosed with diabetes (1: diagnosed with diabetes, 0: not diagnosed with diabetes).

Question 1

Load the data contained in the diabetes.csv file in R.

Solution:

```
file_path <- 'D:/Downloads/diabetes (2).csv'
df <- read_csv(file_path)
head(df)
```

```
## # A tibble: 6 x 9
##   Pregnancies Glucose BloodPressure SkinThickness Insulin   BMI
##         <dbl>   <dbl>         <dbl>         <dbl>   <dbl> <dbl>
## 1           6     148            72            35       0  33.6
## 2           1      85            66            29       0  26.6
## 3           8     183            64             0       0  23.3
## 4           1      89            66            23      94  28.1
## 5           0     137            40            35     168  43.1
## 6           5     116            74             0       0  25.6
## # i 3 more variables: DiabetesPedigreeFunction <dbl>, Age <dbl>, Outcome <dbl>
```

Question 2

Replicate the logic used in the class7.r file to divide the data in a train, validation and test set. Use a 40% - 30% - 30% split.

Solution:

```
set.seed(0)
# train for 40%
is_train <- as.logical(rbinom(dim(df)[1], 1, 0.4))
diabetes_train <- df[is_train, ]
# validation for 30%
```

```
is_validation <- as.logical(rbinom(dim(df)[1], 1, 0.5)* !is_train)
diabetes_validation <- df[is_validation, ]
#test for 30%
diabetes_test <- df[!(is_train|is_validation), ]
```

Question 3

Using all available predictors, fit to the training set: • a classifier based on logistic regression • an LDA classifier • a QDA classifier • a Naive Bayes classifier.

Solution:

```
logistic_diabetes<-glm(Outcome ~., family = "binomial", data=diabetes_train)
# Build the LDA, QDA, and Naive Bayes model.
lda_diabetes_split <- lda(Outcome ~., data = diabetes_train)
qda_diabetes_split <- qda(Outcome ~ ., data = diabetes_train)
nb_diabetes_split <- naiveBayes(Outcome ~ ., data = diabetes_train)
```

Question 4

A group of physician asks you to produce a classifier that achieves 85% Sensitivity when used to test new Pima Indian female patients for diabetes. Using the validation set • plot the ROC curves for the models you built • use the roc function of the pROC library to find - for each of the models you built - the largest threshold t that makes your model achieve at least 90% Sensitivity (just in case, we build some extra margin here to stay a little conservative and make it more likely that we can hit the target Sensitivity goal) • which model performs best (i.e., achieves the largest Specificity) under these conditions?

Solution:

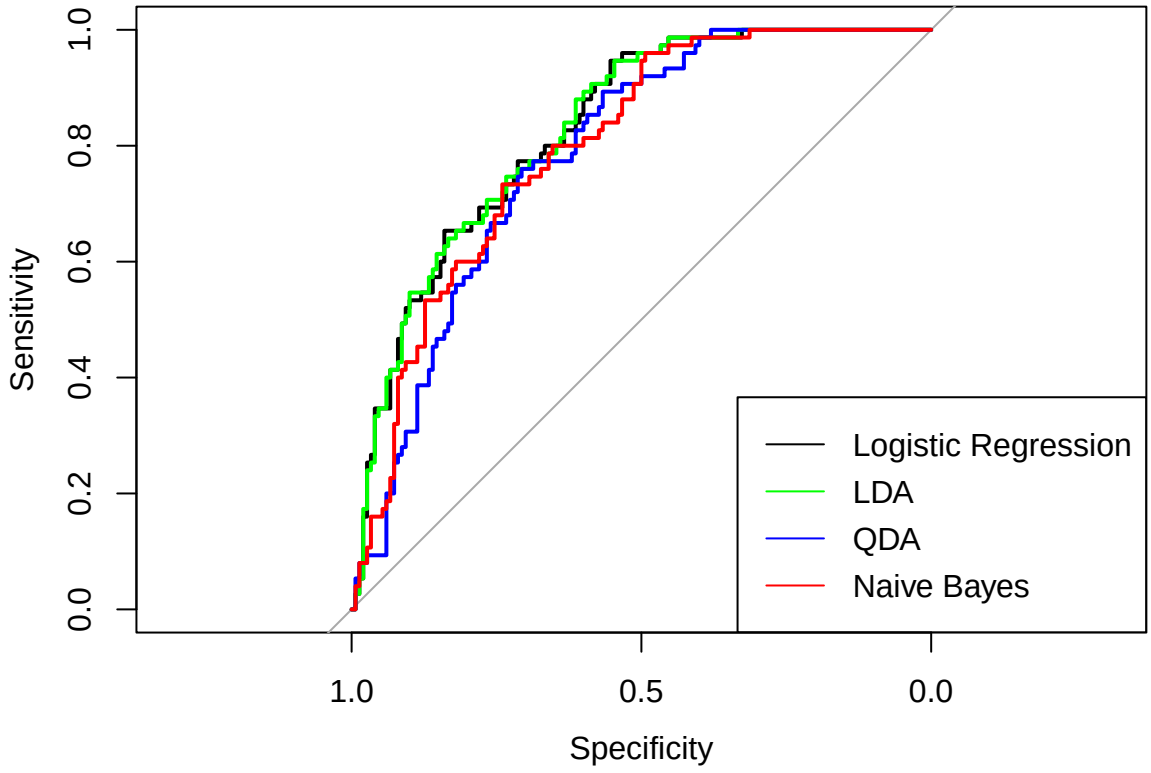
```
# plot the ROC curves for the models you built
plot.roc(diabetes_validation$Outcome,predict(logistic_diabetes, diabetes_validation))

plot.roc(
diabetes_validation$Outcome,
predict(lda_diabetes_split, diabetes_validation)$posterior[, 2], col = "green",
add = TRUE,
)

plot.roc(
diabetes_validation$Outcome,
predict(qda_diabetes_split, diabetes_validation)$posterior[, 2], col = "blue",
add = TRUE,
)

plot.roc(
diabetes_validation$Outcome,
predict(nb_diabetes_split, diabetes_validation, type = "raw")[, 2], col = "red",
add = TRUE,
)

legend("bottomright", legend = c("Logistic Regression", "LDA", "QDA", "Naive Bayes"),
       col = c("black", "green", "blue", "red"), lty = 1)
```




```
# set target sensitivity for logistic
target_sensitivity <- 0.90
# calculate the ROC curve
logistic_diabetes_roc <- roc(
diabetes_validation$Outcome,
predict(logistic_diabetes, diabetes_validation, type = "response") )

# find the largest threshold t that achieves the target sensitivity
logistic_diabetes_roc_index <- ( which.max(logistic_diabetes_roc$sensitivities < target_sensitivity) - 1
logistic_diabetes_t <- logistic_diabetes_roc$threshold[ logistic_diabetes_roc_index
]
# find the specificity of the model at this threshold
logistic_diabetes_roc$specificities[logistic_diabetes_roc_index]
```

```
## [1] 0.58
```

```
logistic_diabetes_t
```

```
## [1] 0.2679169
```

```
# set target sensitivity for lda
target_sensitivity <- 0.90
# calculate the ROC curve
lda_diabetes_roc <- roc(
```

```
diabetes_validation$Outcome,
predict(lda_diabetes_split, diabetes_validation, type = "response")$posterior[,2] )


# find the largest threshold t that achieves the target sensitivity
lda_diabetes_roc_index <- ( which.max(lda_diabetes_roc$sensitivities < target_sensitivity) - 1 )
lda_diabetes_t <- lda_diabetes_roc$threshold[ lda_diabetes_roc_index
]
# find the specificity of the model at this threshold
lda_diabetes_roc$specificities[lda_diabetes_roc_index]
```

```
## [1] 0.5866667
```

```
lda_diabetes_t
```

```
## [1] 0.2670433
```

```
# set target sensitivity for qda
target_sensitivity <- 0.90
# calculate the ROC curve
qda_diabetes_roc <- roc(
diabetes_validation$Outcome,
predict(qda_diabetes_split, diabetes_validation, type = "response")$posterior[,2] )


# find the largest threshold t that achieves the target sensitivity
qda_diabetes_roc_index <- ( which.max(qda_diabetes_roc$sensitivities < target_sensitivity) - 1 )
qda_diabetes_t <- qda_diabetes_roc$threshold[ qda_diabetes_roc_index
]
# find the specificity of the model at this threshold
qda_diabetes_roc$specificities[qda_diabetes_roc_index]
```

```
## [1] 0.5333333
```

```
qda_diabetes_t
```

```
## [1] 0.1256164
```

```
# set target sensitivity for nb
target_sensitivity <- 0.90
# calculate the ROC curve
nb_diabetes_roc <- roc(
diabetes_validation$Outcome,
predict(nb_diabetes_split, diabetes_validation, type = "raw")[,2] )

# find the largest threshold t that achieves the target sensitivity
nb_diabetes_roc_index <- ( which.max(nb_diabetes_roc$sensitivities < target_sensitivity) - 1 )
nb_diabetes_t <- nb_diabetes_roc$threshold[
nb_diabetes_roc_index
]
# find the specificity of the model at this threshold
nb_diabetes_roc$specificities[nb_diabetes_roc_index]
```

```
## [1] 0.5133333
```

```
nb_diabetes_t
```

```
## [1] 0.1169667
```

The LDA model outperforms the QDA, Naive Bayes, and Logistic Regression models, demonstrating the highest specificity score of 0.5866667 under these conditions. This indicates that, in this context, the LDA model stands out as the most successful among the evaluated models.

Question 5

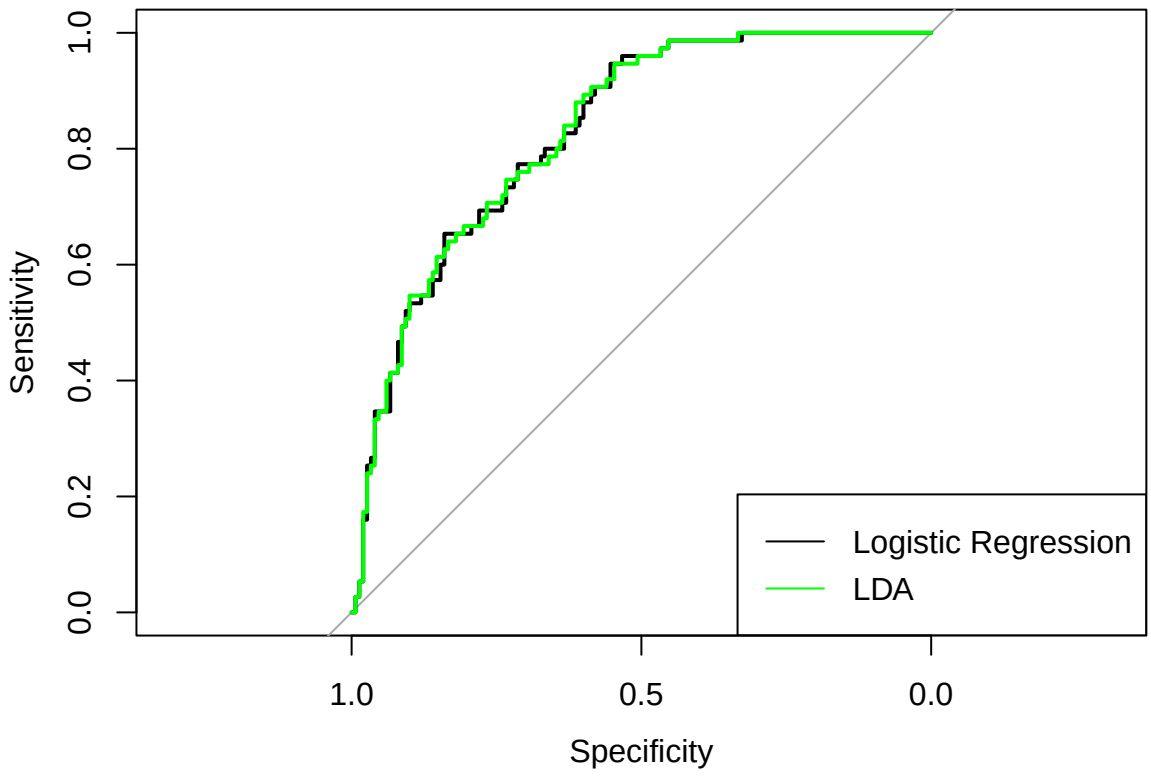
How different are the ROC curves of the classifier obtained by means of logistic regression and of the LDA classifier? Are you surprised by this result? Explain.

Solution:

```
plot.roc(diabetes_validation$Outcome,predict(logistic_diabetes, diabetes_validation))

plot.roc(
diabetes_validation$Outcome,
predict(lda_diabetes_split, diabetes_validation)$posterior[, 2], col = "green",
add = TRUE,
)

legend("bottomright", legend = c("Logistic Regression", "LDA"), col = c("black", "green"), lty = 1)
```



It is not unexpected to observe a similarity between the two curves. The comparable nature of the ROC curves can be attributed to the similar decision boundaries generated by both logistic regression and LDA. This resemblance arises from the shared assumptions of these models, specifically their consideration of a linear relationship between characteristics and the log-odds of class membership.

Question 6

Evaluate the winner model of Question 4 on the test set using the confusionMatrix function. You will need to use the threshold that you computed for this model in Question 4. Does this model seem to satisfy the Sensitivity requirement that the physicians shared with you?

Solution:

```
lda_matrix<- confusionMatrix(
as.factor( ifelse(
predict(lda_diabetes_split, diabetes_test)$posterior[, 2] > lda_diabetes_t, 1,
0
) ),
as.factor(diabetes_test$Outcome), positive = "1",
    mode = "everything"
    )
lda_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 106  11
##          1  58  67
##
##                Accuracy : 0.7149
##                  95% CI : (0.6535, 0.7709)
##     No Information Rate : 0.6777
##     P-Value [Acc > NIR] : 0.1205
##
##                   Kappa : 0.4364
##
##  Mcnemar's Test P-Value : 3.064e-08
##
##             Sensitivity : 0.8590
##             Specificity : 0.6463
##          Pos Pred Value : 0.5360
##          Neg Pred Value : 0.9060
##               Precision : 0.5360
##                  Recall : 0.8590
##                      F1 : 0.6601
##              Prevalence : 0.3223
##          Detection Rate : 0.2769
##    Detection Prevalence : 0.5165
##       Balanced Accuracy : 0.7527
##
##        'Positive' Class : 1
##
```

Yes Indeed, it meets the specified requirement of 85% Sensitivity. The model achieves a senstivity of 85.9%.

Question 7

What is your best estimate about the Specificity that your model will achieve on future patients?

Solution:

```
lda_specificity <- lda_matrix$byClass['Specificity']
lda_specificity
```

```
## Specificity
##   0.6463415
```

The Specificity will be 64.63% from LDA confusion matrix.

```
# to re-verify other models
logistic_matrix <- confusionMatrix( as.factor(
ifelse(
predict(logistic_diabetes, diabetes_test, type = "response") > logistic_diabetes_t,
1,0
) ),
as.factor(diabetes_test$Outcome), positive = "1",
    mode = "everything"
)
logistic_specificity <- logistic_matrix$byClass['Specificity']
logistic_specificity
```

```
## Specificity
##   0.6158537
```

```
qda_matrix <- confusionMatrix( as.factor(
ifelse(
predict(qda_diabetes_split, diabetes_test, type="response")$posterior[, 2] > qda_diabetes_t,
1,0
) ),
as.factor(diabetes_test$Outcome), positive = "1",
    mode = "everything"
)
qda_specificity <- qda_matrix$byClass['Specificity']
qda_specificity
```

```
## Specificity
##   0.5426829
```

```
nb_matrix <- confusionMatrix( as.factor(
ifelse(
predict(nb_diabetes_split, diabetes_test, type = "raw")[,2] > nb_diabetes_t, 1,
0
) ),
as.factor(diabetes_test$Outcome), positive = "1",
    mode = "everything"
)
nb_specificity <- nb_matrix$byClass['Specificity']
nb_specificity
```

```
## Specificity
##   0.4878049
```

Question 8

Fit a knn classifier to the training data and tune the parameter k of your knn classifier using the validation set in such a way that k maximizes the Sensitivity of the classifier on the validation set. You can look back at the class7.r file that we discussed in class and adapt the code from there.

Solution:

```
standardize <- function(x, mean, sd) {
    return((x - mean) / sd)
}

quant_pred_names <- c ("Pregnancies","Glucose","BloodPressure","SkinThickness",
                       "Insulin","BMI","DiabetesPedigreeFunction","Age")

# Means and standard deviations computed on the training data.
mean_train <- sapply(diabetes_train[quant_pred_names], mean)
sd_train <- sapply(diabetes_train[quant_pred_names], sd)

diabetes_train_std <- diabetes_train
diabetes_validation_std <- diabetes_validation
diabetes_test_std <- diabetes_test

# standardize the relevant columns across all 3 splits.
diabetes_train_std[quant_pred_names]<- mapply(
    standardize,
    diabetes_train_std[quant_pred_names],
    mean = mean_train,
    sd = sd_train
)
diabetes_validation_std[quant_pred_names] <- mapply(
    standardize,
    diabetes_validation_std[quant_pred_names],
    mean = mean_train,
    sd = sd_train
)
diabetes_test_std[quant_pred_names] <- mapply(
    standardize,
    diabetes_test_std[quant_pred_names],
    mean = mean_train,
    sd = sd_train
)
```

```
k_candidates <- 2:75
knn_models <- list()
knn_models_performance <- list()
performance_metric <- "Sensitivity"


 for (k in k_candidates) {
    model <- knn(
                diabetes_train_std[quant_pred_names],
```

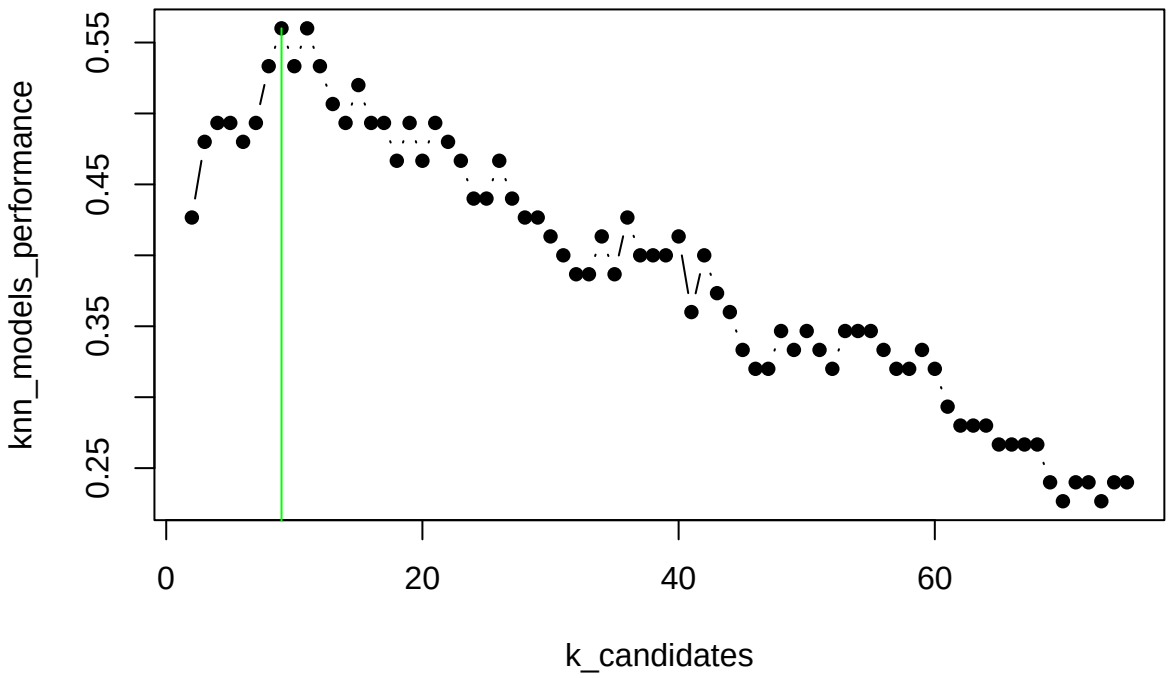
```
            diabetes_validation_std[quant_pred_names],
            diabetes_train_std$Outcome,
              k = k
          )
    performance <- confusionMatrix(
        model,
        as.factor(diabetes_validation_std$Outcome),
        positive = "1",
    mode = "everything"
    )$byClass[performance_metric]
    knn_models <- append(knn_models, model)
    knn_models_performance <- append(knn_models_performance, performance)
}
```

Question 9

What is the best value of k on these data based on your tuning?

Solution:

```
plot(k_candidates, knn_models_performance, type = "b", pch = 16)
best_index <- which.max(knn_models_performance)
best_k <- k_candidates[best_index]
segments(
    best_k,
    0,
    best_k,
    knn_models_performance[[best_index]],
    col = "green"
)
```




```
best_k
```

```
## [1] 9
```

The best k will be 9.

Question 10

Evaluate the knn model on the test set using the confusionMatrix function. Does this knn model perform better or worse than the winner model of Question 4? Which model will you share with the physician to help them diagnose diabetes on future female Pima Indian patients?

```
#KNN model
knn_diabetes_best <- knn( diabetes_train_std[quant_pred_names],
diabetes_test_std[quant_pred_names], diabetes_train_std$Outcome,
k = best_k )
confusionMatrix(
knn_diabetes_best, as.factor(diabetes_test_std$Outcome), positive = "1",
mode = "everything"
)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 143  35
```

```
##          1  21  43
##
##                Accuracy : 0.7686
##                  95% CI : (0.7103, 0.8202)
##     No Information Rate : 0.6777
##     P-Value [Acc > NIR] : 0.001206
##
##                   Kappa : 0.4441
##
##  Mcnemar's Test P-Value : 0.082352
##
##             Sensitivity : 0.5513
##             Specificity : 0.8720
##          Pos Pred Value : 0.6719
##          Neg Pred Value : 0.8034
##               Precision : 0.6719
##                  Recall : 0.5513
##                      F1 : 0.6056
##              Prevalence : 0.3223
##          Detection Rate : 0.1777
##    Detection Prevalence : 0.2645
##       Balanced Accuracy : 0.7116
##
##        'Positive' Class : 1
##
```

```
# winner model in Q4
lda_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 106  11
##          1  58  67
##
##                Accuracy : 0.7149
##                  95% CI : (0.6535, 0.7709)
##     No Information Rate : 0.6777
##     P-Value [Acc > NIR] : 0.1205
##
##                   Kappa : 0.4364
##
##  Mcnemar's Test P-Value : 3.064e-08
##
##             Sensitivity : 0.8590
##             Specificity : 0.6463
##          Pos Pred Value : 0.5360
##          Neg Pred Value : 0.9060
##               Precision : 0.5360
##                  Recall : 0.8590
##                      F1 : 0.6601
##              Prevalence : 0.3223
##          Detection Rate : 0.2769
```

```
##    Detection Prevalence : 0.5165
##       Balanced Accuracy : 0.7527
##
##        'Positive' Class : 1
##
```

Yes,Knn model performs better in terms of specificity and accuracy than the winner model in Question4. While it is true that the KNN model outperforms the winner model from Question 4 in terms of specificity (87.2%) and accuracy (78.86%) with values higher than those of the LDA model (specificity: 64.63%, accuracy: 71.49%), I would still recommend the use of the LDA model for diagnosing diabetes. The primary basis for this recommendation is the high sensitivity of the LDA model, which stands at 85.90%. Given that the aim objective is to achieve a sensitivity of 85%, the LDA model aligns more closely with this crucial criterion. Therefore, despite KNN's better specificity and accuracy, the LDA model is better suited to meet the specific diagnostic goals outlined.