BIOST 2155: Assignment 2 — Solutions

Answer key

Released: 2026-09-25. **Due by**: 2026-10-09, 11:59 PM

Instructions

```
knitr::opts_chunk$set(echo = TRUE, eval = TRUE,
                      fig.height = 4, fig.width = 7, fig.align = "center",
                      message = FALSE, warning = FALSE)
```

This assignment covers **Regression II** (non-linear regression, splines, GAMs) and **Classification I** (logistic regression, generative models). It uses two data sets: `Boston Housing` (`assign1_regData.csv`) for Part A, and `Pima Diabetes` (`assign2_diabetes.csv`) for Part B.

Theory (12 points)

T1. Continuity and smoothness of cubic splines (12 points)

ISLR Problem 7.9.1, page 321. A cubic regression spline with one knot at ξ can be obtained using a basis of the form $x, x^2, x^3, (x-\xi)^3_+$, where $(x-\xi)^3_+ = (x-\xi)^3$ if $x > \xi$ and equals 0 otherwise. We will now show that a function of the form

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\xi)^3_+$$

is indeed a cubic regression spline, regardless of the values of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$.

(a) Find a cubic polynomial $f_1(x) = a_1 + b_1 x + c_1 x^2 + d_1 x^3$ such that $f(x) = f_1(x)$ for all $x \le \xi$. Express a_1, b_1, c_1, d_1 in terms of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$.

(b) Find a cubic polynomial $f_2(x) = a_2 + b_2 x + c_2 x^2 + d_2 x^3$ such that $f(x) = f_2(x)$ for all $x > \xi$. Express a_2, b_2, c_2, d_2 in terms of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$. We have now established that $f(x)$ is a piecewise polynomial.

(c) Show that $f_1(\xi) = f_2(\xi)$, i.e. that $f(x)$ is continuous at ξ.

(d) Show that $f_1'(\xi) = f_2'(\xi)$, i.e. that $f'(x)$ is continuous at ξ.

(e) Show that $f_1''(\xi) = f_2''(\xi)$, i.e. that $f''(x)$ is continuous at ξ.

Together, (c), (d), and (e) establish that $f(x)$ is indeed a cubic spline.

We have $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\xi)^3_+$.

(a) For $x \le \xi$, the term $(x-\xi)^3_+ = 0$, so

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 = f_1(x),$$

giving $a_1 = \beta_0,\ b_1 = \beta_1,\ c_1 = \beta_2,\ d_1 = \beta_3$.

(b) For $x > \xi$, $(x-\xi)_+^3 = (x-\xi)^3 = x^3 - 3\xi x^2 + 3\xi^2 x - \xi^3$. Substituting and collecting powers of x:

$$\begin{aligned} f(x) &= \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 \left(x^3 - 3\xi x^2 + 3\xi^2 x - \xi^3\right) \\ &= (\beta_0 - \beta_4 \xi^3) + (\beta_1 + 3\beta_4 \xi^2)\, x + (\beta_2 - 3\beta_4 \xi)\, x^2 + (\beta_3 + \beta_4)\, x^3 = f_2(x), \end{aligned}$$

so $a_2 = \beta_0 - \beta_4 \xi^3$, $b_2 = \beta_1 + 3\beta_4 \xi^2$, $c_2 = \beta_2 - 3\beta_4 \xi$, $d_2 = \beta_3 + \beta_4$. Since f equals a cubic on each side of ξ, it is a piecewise polynomial.

(c) Continuity of f. $f_2(\xi) - f_1(\xi) = (a_2 - a_1) + (b_2 - b_1)\xi + (c_2 - c_1)\xi^2 + (d_2 - d_1)\xi^3$. Substituting the coefficient differences:

$$= -\beta_4 \xi^3 + 3\beta_4 \xi^2 \cdot \xi - 3\beta_4 \xi \cdot \xi^2 + \beta_4 \xi^3 = \beta_4(-\xi^3 + 3\xi^3 - 3\xi^3 + \xi^3) = 0.$$

So $f_1(\xi) = f_2(\xi)$: f is continuous at ξ.

(d) Continuity of f'. $f_i'(x) = b_i + 2c_i x + 3d_i x^2$, so

$$f_2'(\xi) - f_1'(\xi) = (b_2 - b_1) + 2(c_2 - c_1)\xi + 3(d_2 - d_1)\xi^2 = 3\beta_4 \xi^2 - 6\beta_4 \xi \cdot \xi + 3\beta_4 \xi^2 = 0.$$

So f' is continuous at ξ.

(e) Continuity of f''. $f_i''(x) = 2c_i + 6d_i x$, so

$$f_2''(\xi) - f_1''(\xi) = 2(c_2 - c_1) + 6(d_2 - d_1)\xi = -6\beta_4 \xi + 6\beta_4 \xi = 0.$$

So f'' is continuous at ξ. Together (c)–(e) show f is continuous with continuous first and second derivatives at the knot — i.e. a cubic spline. ■

Application, Part A: Regression II — Non-linear modeling (38 points)

Dataset: `Boston Housing`, saved as `assign1_regData.csv` (same data as Assignment 1).

```
pacman::p_load(tidyverse, caret, splines, gam)
set.seed(42)
data <- read_csv("assign1_regData.csv") |> drop_na()
data$CHAS <- as.factor(data$CHAS)

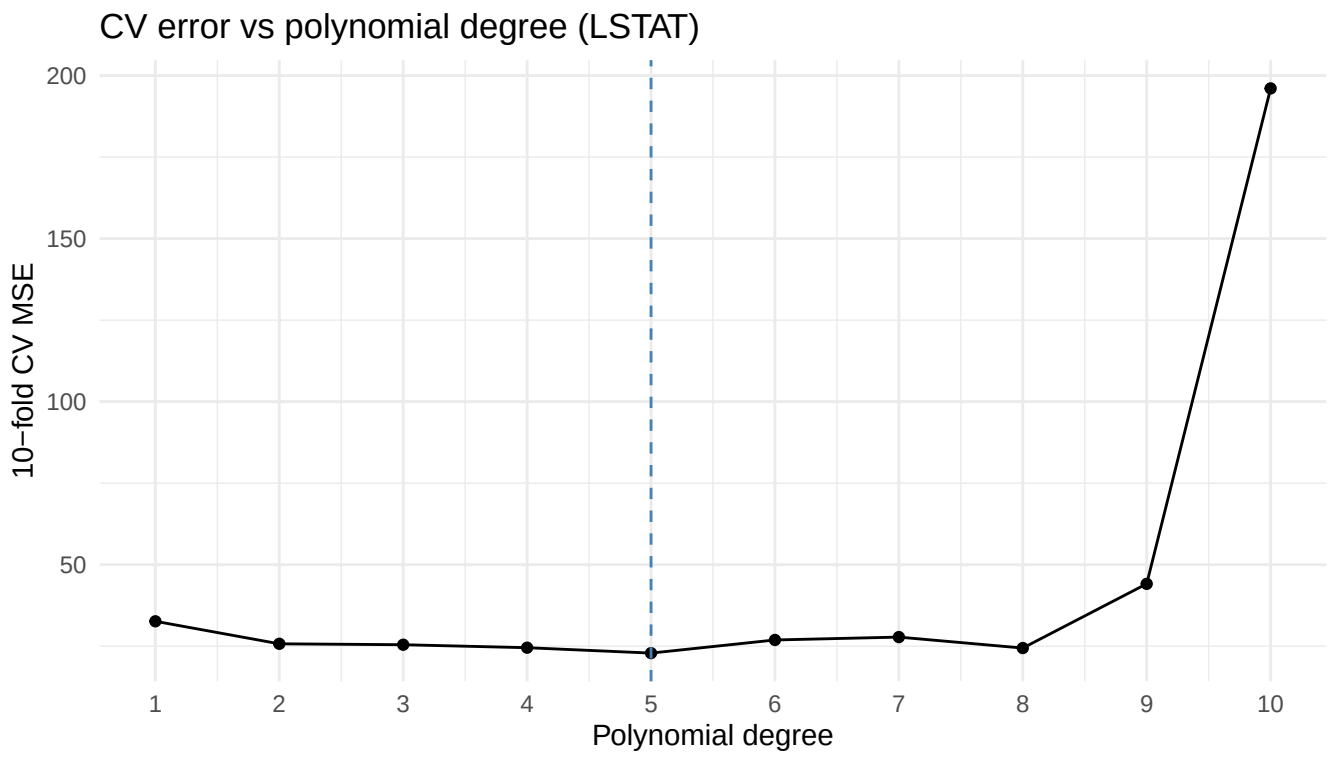
train_index <- createDataPartition(data$MEDV, p = 0.8, list = FALSE)
train <- data[train_index, ]
test  <- data[-train_index, ]
```

B1. Polynomial Regression (5 + 5 + 5 = 15 points)

1. Consider a polynomial regression with `LSTAT` as the only predictor. Use 10-fold CV on the training data to select the optimal polynomial degree (1 to 10). Plot how CV error varies with polynomial degree.

```
set.seed(42)
folds <- createFolds(train$MEDV, k = 10)
cv_err <- sapply(1:10, function(d) {
  mean(sapply(folds, function(idx) {
    fit <- lm(MEDV ~ poly(LSTAT, d), data = train[-idx, ])
    mean((train$MEDV[idx] - predict(fit, train[idx, ]))^2)
  }))
})
best_d <- which.min(cv_err)
```

```
ggplot(tibble(degree = 1:10, cv_mse = cv_err), aes(degree, cv_mse)) +
  geom_line() + geom_point() +
  geom_vline(xintercept = best_d, linetype = 2, colour = "steelblue") +
  scale_x_continuous(breaks = 1:10) +
  labs(x = "Polynomial degree", y = "10-fold CV MSE",
       title = "CV error vs polynomial degree (LSTAT)") +
  theme_minimal()
```



CV error falls through the low degrees, bottoms out at degree 5 here, then grows sharply (degree 10 is catastrophic — the classic polynomial tail blow-up). The exact minimum depends on the random fold split: with different folds a nearby degree (e.g. 4, or 8) can come out lowest. When CV errors are nearly tied, prefer the simpler model — higher-degree polynomials are less stable in the tails.

2. Report the optimal degree and the test set RMSE for that model.

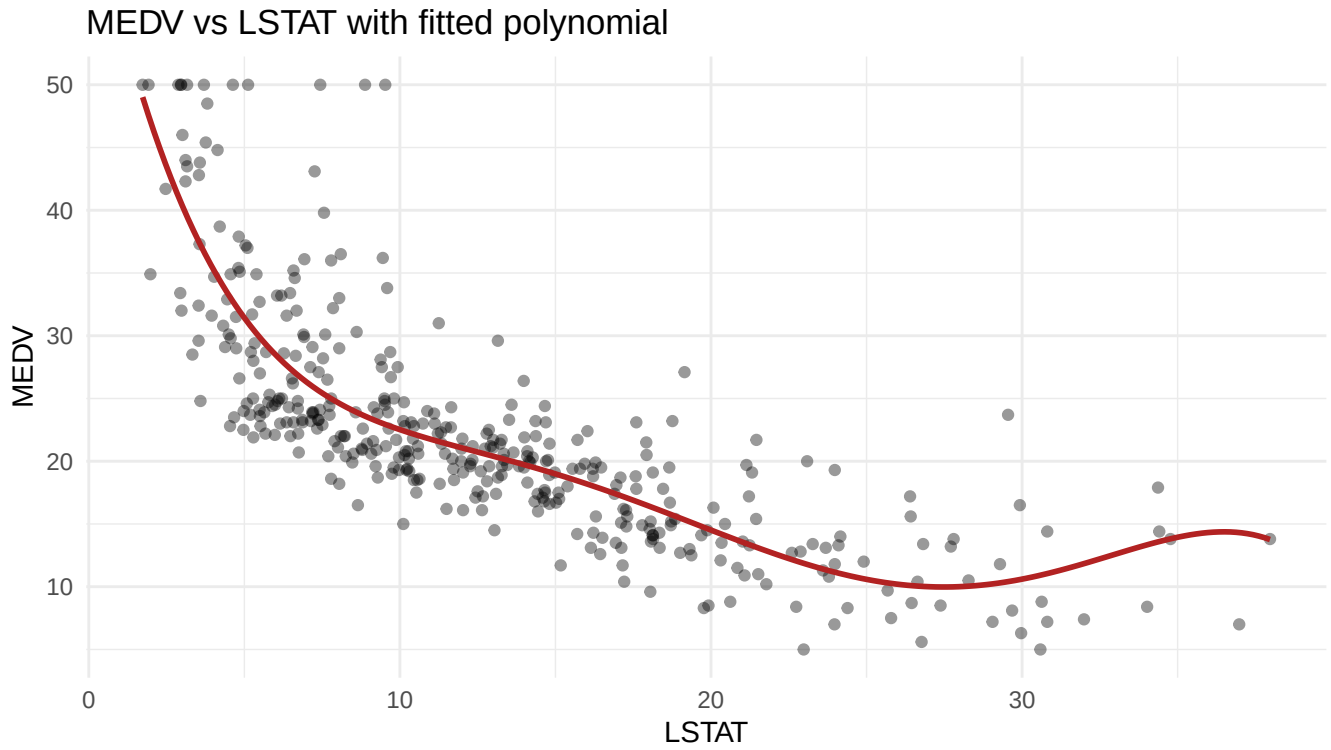
```
fit_poly <- lm(MEDV ~ poly(LSTAT, best_d), data = train)
rmse_poly <- sqrt(mean((test$MEDV - predict(fit_poly, test))^2))
c(optimal_degree = best_d, test_RMSE = round(rmse_poly, 3))
```

```
## optimal_degree      test_RMSE
##          5.000          6.792
```

With degree 5, the test RMSE is about **6.79**.

3. Plot the fitted polynomial curve over a scatterplot of `MEDV` vs `LSTAT`. Does the relationship look like it needed a non-linear fit, or would a straight line have done nearly as well?

```
grid <- tibble(LSTAT = seq(min(data$LSTAT), max(data$LSTAT), length.out = 200))
grid$fit <- predict(fit_poly, grid)
ggplot(data, aes(LSTAT, MEDV)) +
  geom_point(alpha = 0.4) +
  geom_line(data = grid, aes(LSTAT, fit), colour = "firebrick", linewidth = 1) +
  labs(title = "MEDV vs LSTAT with fitted polynomial") + theme_minimal()
```



The relationship is clearly **curved** — MEDV drops steeply as LSTAT rises from low values, then flattens. A straight line would systematically over- and under-predict at the extremes, so the non-linear fit is justified.

B2. Splines (5 + 5 = 10 points)

1. Fit a natural spline (natural spline = cubic spline + linearity constraint in the tails) with `RM` as the predictor, using 4 degrees of freedom. Report the test set RMSE.

```
fit_ns  <- lm(MEDV ~ ns(RM, df = 4), data = train)
rmse_ns <- sqrt(mean((test$MEDV - predict(fit_ns, test))^2))
round(rmse_ns, 3)
```

```
## [1] 6.978
```

2. Compare this test RMSE to a simple linear fit of `MEDV` on `RM` alone. Did the spline improve prediction? Briefly explain, in terms of the shape of the `RM`-`MEDV` relationship, why or why not.

```
fit_lin  <- lm(MEDV ~ RM, data = train)
rmse_lin <- sqrt(mean((test$MEDV - predict(fit_lin, test))^2))
c(natural_spline = round(rmse_ns, 3), linear = round(rmse_lin, 3))
```

```
## natural_spline         linear
##          6.978          7.254
```

The spline (RMSE ≈ 6.98) beats the straight line (RMSE ≈ 7.25). The `RM`–`MEDV` relationship is convex: prices rise slowly across the middle range of room counts and then much faster for large `RM`. A single line cannot bend to follow that curvature, so the flexible spline predicts better.

B3. Generalized Additive Model (GAM) (8 + 5 = 13 points)

1. Fit a GAM for `MEDV` using natural splines for both `LSTAT` and `RM` (4 df each), plus `CHAS` as a linear term. Report the test set RMSE.

```
fit_gam  <- lm(MEDV ~ ns(LSTAT, 4) + ns(RM, 4) + CHAS, data = train)
rmse_gam <- sqrt(mean((test$MEDV - predict(fit_gam, test))^2))
round(rmse_gam, 3)
```

```
## [1] 5.405
```

2. Compare the test RMSE of this GAM to the polynomial model from B1 and the spline model from B2. Which model would you recommend, and why?

```
tibble(model = c("Polynomial (B1)", "Spline RM (B2)", "GAM (B3)"),
       test_RMSE = round(c(rmse_poly, rmse_ns, rmse_gam), 3))
```

```
## # A tibble: 3 x 2
##   model           test_RMSE
##   <chr>               <dbl>
## 1 Polynomial (B1)      6.79
## 2 Spline RM (B2)       6.98
## 3 GAM (B3)             5.40
```

The GAM has the lowest test RMSE (about **5.4**). It combines the two strongest single predictors, each with a flexible smooth term, plus the `CHAS` indicator. I would **recommend the GAM**: it predicts best while staying interpretable — each term's contribution can be examined separately, unlike a single high-degree polynomial.

Application, Part B: Classification I — Logistic regression and LDA (40 points)

Dataset: `Pima Diabetes`, saved as `assign2_diabetes.csv`.

This is a classic dataset of `n = 768` women of Pima Indian heritage, used to predict the onset of diabetes within five years based on diagnostic measurements. Your goal is to build and compare two classifiers: logistic regression (a discriminative model) and linear discriminant analysis (a generative model). **Please set a random seed (`set.seed(42)`) before any step involving data splitting or randomization.**

- `Pregnancies`: Number of times pregnant
- `Glucose`: Plasma glucose concentration (2-hour oral glucose tolerance test)
- `BloodPressure`: Diastolic blood pressure (mm Hg)
- `SkinThickness`: Triceps skin fold thickness (mm)
- `Insulin`: 2-Hour serum insulin (mu U/mL)
- `BMI`: Body mass index
- `DiabetesPedigreeFunction`: A function that scores likelihood of diabetes based on family history
- `Age`: Age in years
- `Outcome`: 1 if the patient developed diabetes, 0 otherwise

```
pacman::p_load(tidyverse, caret, MASS, pROC)
set.seed(42)
diab <- read_csv("assign2_diabetes.csv") |> drop_na()
diab$Outcome <- factor(diab$Outcome, levels = c(0, 1), labels = c("No", "Yes"))
```

C1. Data Exploration & Splitting (5 + 5 = 10 points)

1. Use `summary()` to explore the dataset. Comment on any variables that look like they may have implausible zero values (e.g. `Glucose` or `BMI` of zero) and briefly explain why that matters for the analysis (you do not need to fix it — just note the issue).

```
summary(diab)
```

```
##   Pregnancies        Glucose       BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   : 44.00   Min.   : 24.00   Min.   : 7.00
##  1st Qu.: 1.000   1st Qu.: 99.75   1st Qu.: 64.00   1st Qu.:25.00
##  Median : 3.000   Median :117.00   Median : 72.00   Median :28.00
##  Mean   : 3.845   Mean   :121.68   Mean   : 72.39   Mean   :29.09
##  3rd Qu.: 6.000   3rd Qu.:140.25   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.00   Max.   :122.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   : 14.0   Min.   :18.20   Min.   :0.0780           Min.   :21.00
##  1st Qu.:102.5   1st Qu.:27.50   1st Qu.:0.2437           1st Qu.:24.00
##  Median :102.5   Median :32.05   Median :0.3725           Median :29.00
##  Mean   :141.8   Mean   :32.43   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:169.5   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##  Outcome
##  No :500
##  Yes:268
##
##
##
##
```

```
# check the variables where a zero would be physiologically impossible
sapply(c("Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"),
       function(v) sum(diab[[v]] == 0))
```

```
##       Glucose BloodPressure SkinThickness       Insulin           BMI
##             0             0             0             0             0
```

In the *original* Pima dataset, `Glucose`, `BloodPressure`, `SkinThickness`, `Insulin`, and `BMI` contain zeros that are physiologically impossible — a living patient cannot have a BMI or glucose of 0. Those zeros are really missing values coded as 0, and treating them as real numbers would bias every model (a fake cluster of "0" pulls means and covariances toward zero).

Note for this version of the file: the distributed CSV has **already been cleaned** — the zero counts above are all 0, and the repeated `Insulin`/ `SkinThickness` values (e.g. 102.5, 169.5) are imputed constants. So a correct answer here is either the general point about zero-coded missingness, or the observation that the data appear pre-imputed. Either observation is a correct answer.

2. Randomly split the data into a training set (75%) and a test set (25%), stratified by `Outcome` (use `caret::createDataPartition`, `seed = 42`).

```r
set.seed(42)
ti <- createDataPartition(diab$Outcome, p = 0.75, list = FALSE)
tr <- diab[ti, ]; te <- diab[-ti, ]
c(train = nrow(tr), test = nrow(te))
```

```
## train  test
##   576   192
```

C2. Logistic Regression (5 + 5 + 5 = 15 points)

1. Fit a logistic regression model predicting `Outcome` from all eight predictors on the training data. Which predictors are statistically significant at the 0.05 level?

```r
glm_fit <- glm(Outcome ~ ., data = tr, family = binomial)
summary(glm_fit)
```

```
##
## Call:
## glm(formula = Outcome ~ ., family = binomial, data = tr)
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -9.438283   0.962832  -9.803  < 2e-16 ***
## Pregnancies               0.148606   0.037842   3.927 8.60e-05 ***
## Glucose                   0.029815   0.004635   6.432 1.26e-10 ***
## BloodPressure            -0.002614   0.010085  -0.259 0.795507
## SkinThickness             0.045377   0.016086   2.821 0.004789 **
## Insulin                   0.006306   0.001798   3.508 0.000452 ***
## BMI                       0.057358   0.020866   2.749 0.005979 **
## DiabetesPedigreeFunction  0.864915   0.340152   2.543 0.010999 *
## Age                       0.001667   0.011482   0.145 0.884530
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 745.11  on 575  degrees of freedom
## Residual deviance: 510.09  on 567  degrees of freedom
## AIC: 528.09
##
## Number of Fisher Scoring iterations: 5
```

```r
co <- summary(glm_fit)$coefficients
rownames(co)[co[, 4] < 0.05 & rownames(co) != "(Intercept)"]
```

```
## [1] "Pregnancies"              "Glucose"
## [3] "SkinThickness"            "Insulin"
## [5] "BMI"                      "DiabetesPedigreeFunction"
```

Significant predictors are **Pregnancies, Glucose, SkinThickness, Insulin, BMI, and DiabetesPedigreeFunction**. `BloodPressure` and `Age` are not significant. (`Glucose` is the strongest, as expected clinically.)

2. Using a 0.5 probability threshold, generate class predictions on the test set and report the confusion matrix (`caret::confusionMatrix`).

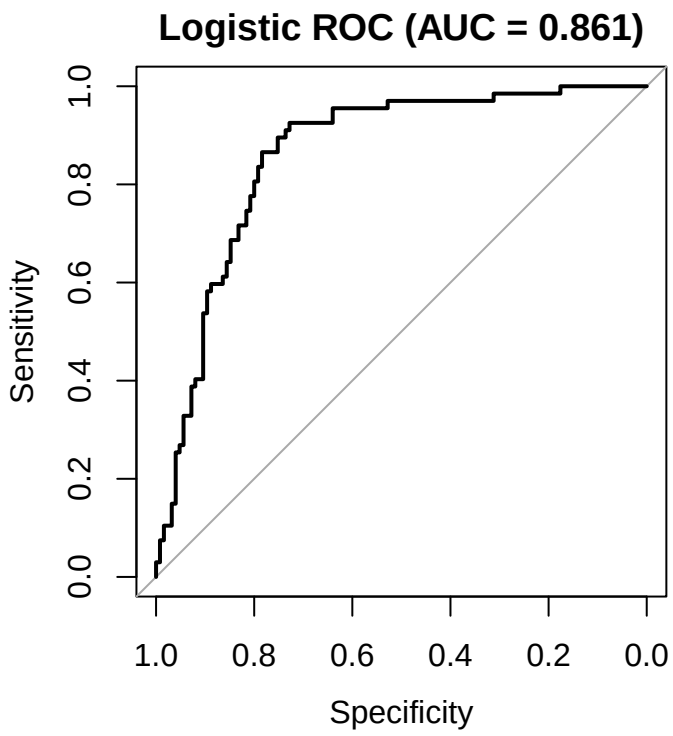
```
p_glm    <- predict(glm_fit, te, type = "response")
pred_glm <- factor(ifelse(p_glm > 0.5, "Yes", "No"), levels = c("No", "Yes"))
cm_glm   <- confusionMatrix(pred_glm, te$Outcome, positive = "Yes")
cm_glm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  108  26
##        Yes  17  41
##
##                Accuracy : 0.776
##                  95% CI : (0.7104, 0.8329)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 0.0001183
##
##                   Kappa : 0.4912
##
##  Mcnemar's Test P-Value : 0.2224692
##
##             Sensitivity : 0.6119
##             Specificity : 0.8640
##          Pos Pred Value : 0.7069
##          Neg Pred Value : 0.8060
##              Prevalence : 0.3490
##          Detection Rate : 0.2135
##    Detection Prevalence : 0.3021
##       Balanced Accuracy : 0.7380
##
##        'Positive' Class : Yes
##
```

Accuracy is about **78%**, with high specificity (≈ 0.86) but lower sensitivity (≈ 0.61) — the model misses a fair share of true diabetics at the 0.5 cut-off.

3. Plot the ROC curve for this model on the test set and report the AUC.

```
roc_glm <- roc(te$Outcome, p_glm, quiet = TRUE)
plot(roc_glm, main = sprintf("Logistic ROC (AUC = %.3f)", auc(roc_glm)))
```




```
auc(roc_glm)
```

```
## Area under the curve: 0.861
```

AUC ≈ **0.86** — good discrimination.

C3. Linear Discriminant Analysis & Comparison (5 + 5 + 5 = 15 points)

1. Fit an LDA model with the same eight predictors on the training data, and generate class predictions on the test set.

```
lda_fit  <- lda(Outcome ~ ., data = tr)
lda_pred <- predict(lda_fit, te)
```

2. Report the LDA confusion matrix and AUC (using the posterior probability of `Yes`) on the test set.

```
cm_lda <- confusionMatrix(lda_pred$class, te$Outcome, positive = "Yes")
cm_lda
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  109  27
##        Yes  16  40
##
##                Accuracy : 0.776
##                  95% CI : (0.7104, 0.8329)
##     No Information Rate : 0.651
```

```
##     P-Value [Acc > NIR] : 0.0001183
##
##                   Kappa : 0.4876
##
##  Mcnemar's Test P-Value : 0.1272627
##
##             Sensitivity : 0.5970
##             Specificity : 0.8720
##          Pos Pred Value : 0.7143
##          Neg Pred Value : 0.8015
##              Prevalence : 0.3490
##          Detection Rate : 0.2083
##    Detection Prevalence : 0.2917
##       Balanced Accuracy : 0.7345
##
##        'Positive' Class : Yes
##
```

```
roc_lda <- roc(te$Outcome, lda_pred$posterior[, "Yes"], quiet = TRUE)
auc(roc_lda)
```

```
## Area under the curve: 0.855
```

3. Interpretation: *Compare logistic regression and LDA on accuracy, sensitivity, specificity, and AUC. LDA assumes predictors are approximately multivariate normal within each class and share a common covariance matrix — do you expect that assumption to hold reasonably well for this data? Which model would you recommend for this problem, and why?*

```
tibble(
  model = c("Logistic", "LDA"),
  accuracy    = c(cm_glm$overall["Accuracy"],       cm_lda$overall["Accuracy"]),
  sensitivity = c(cm_glm$byClass["Sensitivity"],    cm_lda$byClass["Sensitivity"]),
  specificity = c(cm_glm$byClass["Specificity"],    cm_lda$byClass["Specificity"]),
  auc         = c(as.numeric(auc(roc_glm)),         as.numeric(auc(roc_lda)))
) |> mutate(across(-model, \(x) round(x, 3)))
```

```
## # A tibble: 2 x 5
##   model    accuracy sensitivity specificity   auc
##   <chr>       <dbl>       <dbl>       <dbl> <dbl>
## 1 Logistic    0.776       0.612       0.864 0.861
## 2 LDA         0.776       0.597       0.872 0.855
```

The two models perform **almost identically**: the same accuracy (≈ 0.78), similar sensitivity/specificity, and AUCs within about 0.01 (logistic edges LDA).

LDA assumes the predictors are multivariate normal *within each class* and share a *common covariance matrix*. Neither holds cleanly here: several predictors are right-skewed (`Insulin`, `DiabetesPedigreeFunction`, `Pregnancies`), and the diabetic group tends to be more variable than the non-diabetic group, so the equal-covariance assumption is doubtful. That LDA still matches logistic regression shows it is fairly robust to mild violations, but because logistic regression makes **no distributional assumption** on the predictors and does marginally better on AUC, I would **recommend logistic regression** for this problem. Its coefficients also give direct, interpretable odds-ratio statements for clinicians.