Exercise 9

Advanced Methods for Regression and Classification

Rita Selimi

18/12/2024

Data Preparation

```
# Load the Diabetes data
data(Diabetes, package = "ROCit")

# Exclude variables with high missingness
excluded_vars <- c("id", "bp.2s", "bp.2d")
diabetes_data <- Diabetes %>%
    select(-all_of(excluded_vars))

# Impute missing values for `frame` using mode imputation
mode_impute <- function(x) {
    levels(x)[which.max(tabulate(match(x, levels(x))))]
}
diabetes_data$frame[is.na(diabetes_data$frame)] <- mode_impute(diabetes_data$frame)

# Remove rows with any missing values
diabetes_data <- na.omit(diabetes_data)

# Verify the updated dataset
cat("Remaining rows after removing missing values:", nrow(diabetes_data), "\n")
```

```
## Remaining rows after removing missing values: 375
```

```
cat("Remaining columns:", ncol(diabetes_data), "\n")
```

```
## Remaining columns: 19
```

```
summary(diabetes_data)
```

```
##       chol          stab.glu          hdl             ratio
##  Min.   : 78.0   Min.   : 48.0   Min.   : 12.00   Min.   : 1.500
##  1st Qu.:179.0   1st Qu.: 81.0   1st Qu.: 38.00   1st Qu.: 3.200
##  Median :204.0   Median : 90.0   Median : 46.00   Median : 4.200
##  Mean   :207.6   Mean   :107.6   Mean   : 50.43   Mean   : 4.525
##  3rd Qu.:229.5   3rd Qu.:108.5   3rd Qu.: 59.00   3rd Qu.: 5.400
```

```
##  Max.   :443.0   Max.   :385.0   Max.   :120.00   Max.   :19.300
##      glyhb             location       age           gender       height
##  Min.   : 2.680   Buckingham:181   Min.   :19.00   female:220   Min.   :52
##  1st Qu.: 4.395   Louisa    :194   1st Qu.:34.00   male  :155   1st Qu.:63
##  Median : 4.860                    Median :45.00                Median :66
##  Mean   : 5.603                    Mean   :46.98                Mean   :66
##  3rd Qu.: 5.630                    3rd Qu.:60.00                3rd Qu.:69
##  Max.   :16.110                    Max.   :92.00                Max.   :76
##      weight         frame         bp.1s           bp.1d            waist
##  Min.   : 99.0   large : 96   Min.   : 90.0   Min.   : 48.00   Min.   :26.00
##  1st Qu.:151.0   medium:181   1st Qu.:121.5   1st Qu.: 75.00   1st Qu.:33.00
##  Median :174.0   small : 98   Median :136.0   Median : 82.00   Median :37.00
##  Mean   :177.9                Mean   :137.5   Mean   : 83.38   Mean   :37.95
##  3rd Qu.:200.0                3rd Qu.:148.0   3rd Qu.: 91.00   3rd Qu.:41.50
##  Max.   :325.0                Max.   :250.0   Max.   :124.00   Max.   :56.00
##       hip            time.ppn          bmi            dtest
##  Min.   :30.00   Min.   :   5   Min.   :16.00   Length:375
##  1st Qu.:39.00   1st Qu.:  90   1st Qu.:24.12   Class :character
##  Median :42.00   Median : 240   Median :27.79   Mode  :character
##  Mean   :43.09   Mean   : 335   Mean   :28.82
##  3rd Qu.:46.00   3rd Qu.: 480   3rd Qu.:32.26
##  Max.   :64.00   Max.   :1560   Max.   :55.78
##       whr
##  Min.   :0.6818
##  1st Qu.:0.8293
##  Median :0.8800
##  Mean   :0.8811
##  3rd Qu.:0.9259
##  Max.   :1.1429
```

```
# Check for class imbalance in target variable
table(diabetes_data$dtest)
```

```
##
##   -   +
## 317  58
```

```
# Select only numeric predictors
numeric_data <- diabetes_data %>%
    select(-dtest) %>%
    select_if(is.numeric)

# Compute the correlation matrix
cor_matrix <- cor(numeric_data, use = "complete.obs")
cat("Correlation Matrix:\n")
```

```
## Correlation Matrix:
```

```
print(cor_matrix)
```

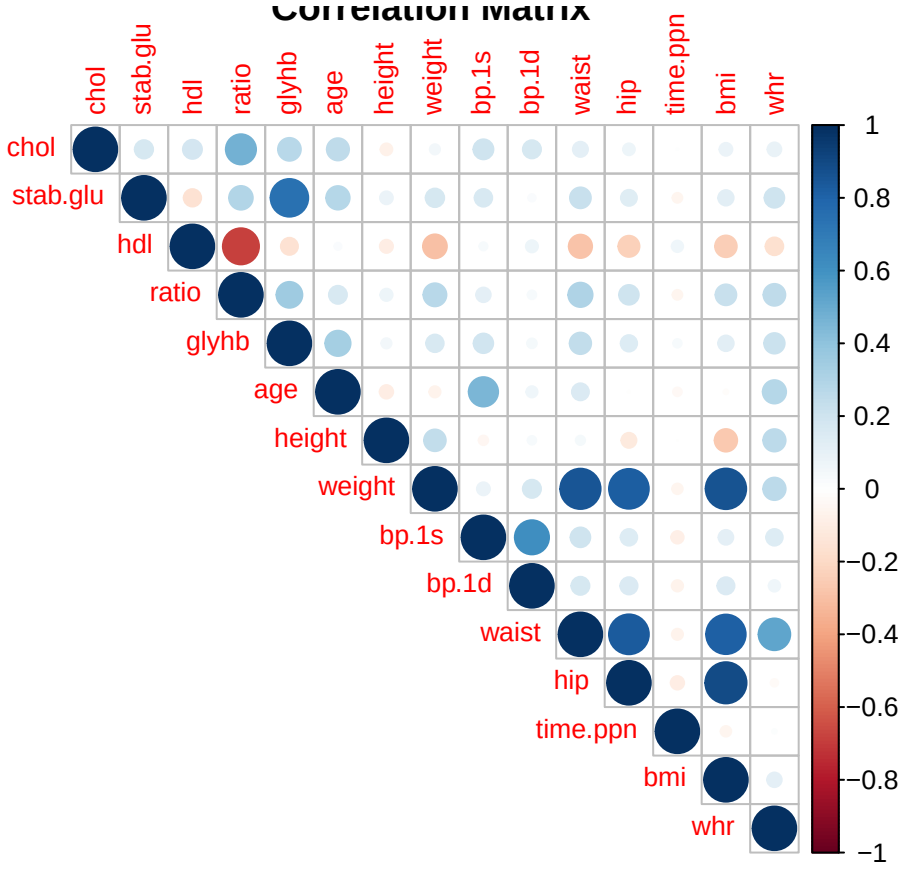
```
##                   chol    stab.glu         hdl       ratio       glyhb
## chol      1.0000000000  0.17605760  0.18918622  0.47685119  0.27387403
```

```
## stab.glu  0.1760576046  1.00000000 -0.15845068  0.29353513  0.74264681
## hdl       0.1891862222 -0.15845068  1.00000000 -0.68325757 -0.15190578
## ratio     0.4768511854  0.29353513 -0.68325757  1.00000000  0.35021489
## glyhb     0.2738740294  0.74264681 -0.15190578  0.35021489  1.00000000
## age       0.2580751516  0.28820616  0.02624555  0.16952389  0.33386994
## height   -0.0726825795  0.08643024 -0.09014271  0.07828072  0.05431960
## weight    0.0545001654  0.17918946 -0.29760400  0.27690932  0.16074457
## bp.1s     0.2084006482  0.16181629  0.03436444  0.11090086  0.19964120
## bp.1d     0.1715635556  0.02867443  0.07630061  0.03904864  0.04906403
## waist     0.1192740454  0.22720469 -0.28499211  0.30894469  0.24116131
## hip       0.0732717715  0.13333204 -0.23280194  0.20246310  0.14093763
## time.ppn  0.0009236077 -0.05010410  0.06822885 -0.05007620  0.03783154
## bmi       0.0868511665  0.12405699 -0.24582837  0.22717762  0.12696185
## whr       0.0993955976  0.20339633 -0.16404526  0.25006311  0.21925187
##                  age        height      weight       bp.1s       bp.1d
## chol      0.25807515 -0.0726825795  0.05450017  0.20840065  0.17156356
## stab.glu  0.28820616  0.0864302351  0.17918946  0.16181629  0.02867443
## hdl       0.02624555 -0.0901427069 -0.29760400  0.03436444  0.07630061
## ratio     0.16952389  0.0782807236  0.27690932  0.11090086  0.03904864
## glyhb     0.33386994  0.0543196045  0.16074457  0.19964120  0.04906403
## age       1.00000000 -0.0953003330 -0.06523800  0.45236549  0.06562279
## height   -0.09530033  1.0000000000  0.24609012 -0.04709996  0.03839032
## weight   -0.06523800  0.2460901156  1.00000000  0.08797282  0.17190112
## bp.1s     0.45236549 -0.0470999583  0.08797282  1.00000000  0.61104157
## bp.1d     0.06562279  0.0383903197  0.17190112  0.61104157  1.00000000
## waist     0.15068806  0.0450335672  0.85176414  0.20342942  0.17056560
## hip      -0.00208207 -0.1176291370  0.82931024  0.14425492  0.15538918
## time.ppn -0.03962976 -0.0001877958 -0.05772871 -0.08143493 -0.06694966
## bmi      -0.01132405 -0.2652468651  0.86195093  0.11438063  0.15287602
## whr       0.28614107  0.2617183760  0.26420206  0.14409846  0.06933993
##                waist         hip      time.ppn         bmi         whr
## chol      0.11927405  0.07327177  0.0009236077  0.08685117  0.09939560
## stab.glu  0.22720469  0.13333204 -0.0501040985  0.12405699  0.20339633
## hdl      -0.28499211 -0.23280194  0.0682288499 -0.24582837 -0.16404526
## ratio     0.30894469  0.20246310 -0.0500762031  0.22717762  0.25006311
## glyhb     0.24116131  0.14093763  0.0378315395  0.12696185  0.21925187
## age       0.15068806 -0.00208207 -0.0396297574 -0.01132405  0.28614107
## height    0.04503357 -0.11762914 -0.0001877958 -0.26524687  0.26171838
## weight    0.85176414  0.82931024 -0.0577287098  0.86195093  0.26420206
## bp.1s     0.20342942  0.14425492 -0.0814349347  0.11438063  0.14409846
## bp.1d     0.17056560  0.15538918 -0.0669496556  0.15287602  0.06933993
## waist     1.00000000  0.83224050 -0.0647235445  0.81805720  0.52613334
## hip       0.83224050  1.00000000 -0.0943397549  0.89118868 -0.02957802
## time.ppn -0.06472354 -0.09433975  1.0000000000 -0.05852466  0.01035406
## bmi       0.81805720  0.89118868 -0.0585246604  1.00000000  0.11107930
## whr       0.52613334 -0.02957802  0.0103540610  0.11107930  1.00000000
```

```
# Visualize the correlation matrix
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
corrplot(cor_matrix, method = "circle", type = "upper", tl.cex = 0.8, main = "Correlation Matrix")
```



The correlation matrix shows strong correlations among **weight**, **waist**, **hip**, and **bmi** (correlation > 0.8), indicating redundancy. Retaining **bmi** is sufficient as it summarizes body size, so the other three can be removed. Similarly, **bp.1s** and **bp.1d** are moderately correlated (**0.61**), so keeping only **bp.1s** is appropriate. Finally, **stab.glu** and **glyhb** are moderately correlated (**0.74**), so one of them can be dropped to simplify the model.

```r
# Identify highly correlated pairs (correlation > 0.7 but < 1)
high_corr <- which(abs(cor_matrix) > 0.7 & abs(cor_matrix) < 1, arr.ind = TRUE)
cat("Highly Correlated Predictor Pairs:\n")
```

```
## Highly Correlated Predictor Pairs:
```

```r
print(high_corr)
```

```
##          row col
## glyhb      5   2
## stab.glu   2   5
## waist     11   8
## hip       12   8
## bmi       14   8
## weight     8  11
## hip       12  11
```

```
## bmi       14  11
## weight     8  12
## waist     11  12
## bmi       14  12
## weight     8  14
## waist     11  14
## hip       12  14
```

```
# Drop redundant predictors Keep 'bmi' (drop 'weight', 'waist', 'hip'), keep
# 'bp.1s' (drop 'bp.1d'), keep 'stab.glu' (drop 'glyhb')
diabetes_data <- diabetes_data %>%
    select(-waist, -hip, -bp.1d, -glyhb, -ratio, -weight)

# Verify remaining predictors
cat("Remaining predictors after removing highly correlated ones:\n")
```

```
## Remaining predictors after removing highly correlated ones:
```

```
print(names(diabetes_data))
```

```
##  [1] "chol"     "stab.glu" "hdl"      "location" "age"      "gender"
##  [7] "height"   "frame"    "bp.1s"    "time.ppn" "bmi"      "dtest"
## [13] "whr"
```

The cleaned dataset now includes 13 predictors, ensuring less redundancy and reducing multicollinearity for building a stable classification model.

```
# Update the training and test splits after removing missing values
set.seed(123)  # For reproducibility
train_index <- createDataPartition(diabetes_data$dtest, p = 0.75, list = FALSE)
train_data <- diabetes_data[train_index, ]
test_data <- diabetes_data[-train_index, ]

# Print the sizes of the training and test sets
cat("Training set size after removing missing values:", nrow(train_data), "\n")
```

```
## Training set size after removing missing values: 282
```

```
cat("Test set size after removing missing values:", nrow(test_data), "\n")
```

```
## Test set size after removing missing values: 93
```

1. Logistic Regression

```
# Ensure dtest is a factor (0 and 1)
train_data$dtest <- ifelse(train_data$dtest == "+", 1, 0)
test_data$dtest <- ifelse(test_data$dtest == "+", 1, 0)
train_data$dtest <- as.factor(train_data$dtest)
test_data$dtest <- as.factor(test_data$dtest)
```

```r
# Fit a logistic regression model using the training set
logistic_model <- glm(dtest ~ ., data = train_data, family = "binomial")

# Summary of the model
cat("Logistic Regression Summary:\n")
```

```
## Logistic Regression Summary:
```

```r
print(summary(logistic_model))
```

```
##
## Call:
## glm(formula = dtest ~ ., family = "binomial", data = train_data)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -4.4884085  8.1733499  -0.549    0.583
## chol            0.0060077  0.0079508   0.756    0.450
## stab.glu        0.0546840  0.0097871   5.587 2.31e-08 ***
## hdl             0.0111542  0.0176597   0.632    0.528
## locationLouisa  0.0325918  0.6396989   0.051    0.959
## age             0.0305584  0.0229953   1.329    0.184
## gendermale     -0.0804470  0.8989465  -0.089    0.929
## height         -0.1563782  0.1105386  -1.415    0.157
## framemedium     0.7921489  0.7580613   1.045    0.296
## framesmall      0.5176599  1.0626157   0.487    0.626
## bp.1s           0.0168129  0.0116350   1.445    0.148
## time.ppn        0.0012980  0.0008221   1.579    0.114
## bmi             0.0445867  0.0495902   0.899    0.369
## whr            -2.0176768  4.6376296  -0.435    0.664
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 244.226  on 281  degrees of freedom
## Residual deviance:  91.725  on 268  degrees of freedom
## AIC: 119.73
##
## Number of Fisher Scoring iterations: 7
```

```r
# Predict probabilities on the test set
test_predictions <- predict(logistic_model, newdata = test_data, type = "response")

# Convert probabilities to class labels (0 or 1)
test_class_predictions <- ifelse(test_predictions > 0.5, 1, 0)

# Confusion matrix
confusion_matrix <- table(Predicted = test_class_predictions, Actual = test_data$dtest)
cat("Confusion Matrix:\n")
```

```
## Confusion Matrix:
```

```r
print(confusion_matrix)
```

```
##           Actual
## Predicted  0  1
##         0 75  6
##         1  4  8
```

```r
# Calculate misclassification rate
misclassification_rate <- mean(test_class_predictions != test_data$dtest)
cat("Misclassification Rate:", misclassification_rate, "\n")
```

```
## Misclassification Rate: 0.1075269
```

A **logistic regression model** is a statistical model used to predict a **binary outcome** (e.g., yes/no, 0/1, success/failure) based on one or more predictor variables.

Problems Observed in the Logistic Regression Model:

1. **Insignificant Predictors**:
   - Many predictors, including `chol`, `hdl`, `locationLouisa`, `gender`, `height`, `frame`, `bp.1s`, `time.ppn`, `bmi`, and `whr`, have **high p-values** (greater than 0.05), indicating they are not statistically significant.
   - Only `stab.glu` (p-value 2.31e-08) is a strong predictor, while `age`, `bp.1s`, and `time.ppn` show some borderline significance.
2. **Potential Underfitting**:
   - The **residual deviance** (91.725) is much lower than the **null deviance**, how much the response variable deviates from its mean without considering any predictors (244.226), indicating the model explains a substantial amount of variation in the data.
   - However, the misclassification rate (10.75%) suggests the model may struggle to generalize well, particularly for identifying the positive class.
3. **Multicollinearity**:
   - Multicollinearity issues appear to have been addressed by removing highly correlated variables earlier in the process. This has likely improved model stability, as coefficients are no longer extreme or inconsistent.
4. **Confusion Matrix and Class Imbalance**:
   - The confusion matrix shows the model performs reasonably well in predicting the negative class (75 true negatives and 4 false positives) but struggles with the positive class (8 true positives and 6 false negatives).
   - The imbalance in identifying the positive class is evident and problematic, especially if the goal is to identify diabetes cases accurately.

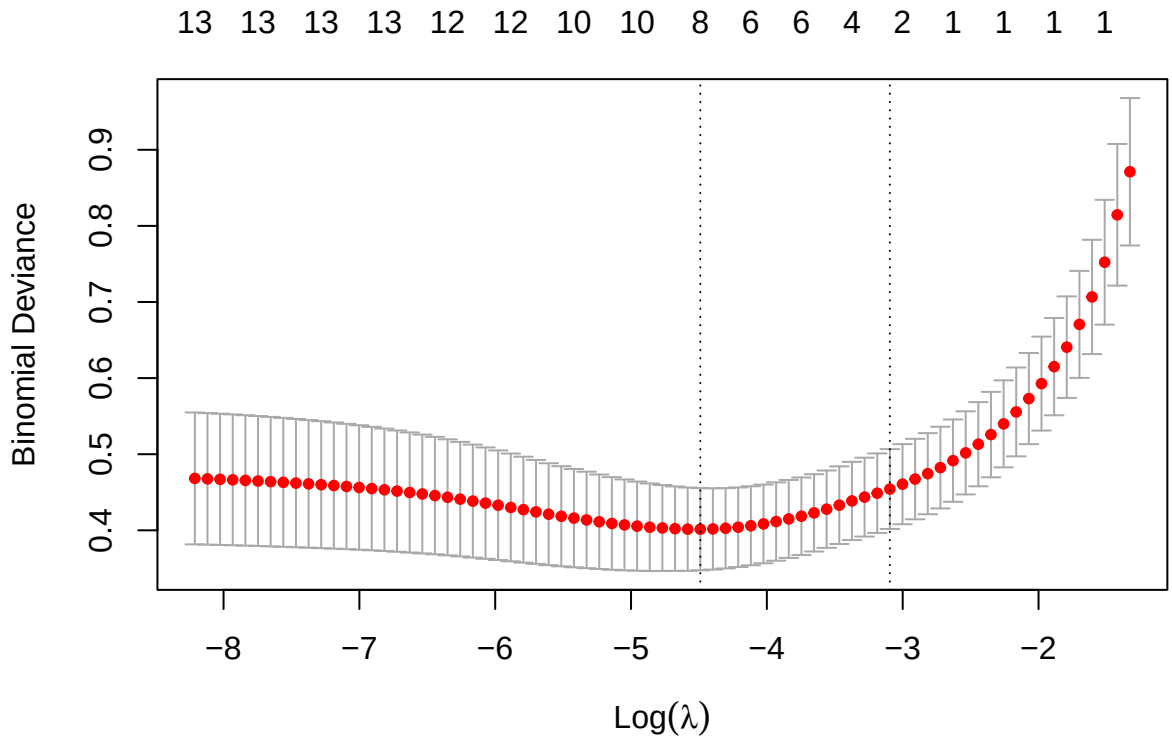
2. Sparse Logistic Regression (Lasso)

```r
# Prepare data for glmnet (requires matrix for predictors)
x_train <- model.matrix(dtest ~ ., data = train_data)[, -1]  # Remove intercept
y_train <- as.numeric(as.character(train_data$dtest))  # Ensure y is numeric (0/1)

x_test <- model.matrix(dtest ~ ., data = test_data)[, -1]
y_test <- as.numeric(as.character(test_data$dtest))  # Ensure test labels are numeric

# Fit Lasso (alpha = 1) with cross-validation
set.seed(123)
# Lasso Regularization (L1), meaning some coefficients are set to zero
cv_lasso <- cv.glmnet(x_train, y_train, family = "binomial", alpha = 1)

# Plot cross-validation curve
plot(cv_lasso)
```




```r
cat("Optimal lambda:", cv_lasso$lambda.min, "\n")
```

```
## Optimal lambda: 0.01122084
```

```r
# Refit the model using optimal lambda
lasso_model <- glmnet(x_train, y_train, family = "binomial", alpha = 1, lambda = cv_lasso$lambda.min)

# Display coefficients
lasso_coef <- coef(lasso_model)
cat("Non-zero coefficients (Lasso):\n")
```

```
## Non-zero coefficients (Lasso):
```

```
print(lasso_coef)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                              s0
## (Intercept)    -2.6483689895
## chol            0.0040424504
## stab.glu        0.0397296685
## hdl             .
## locationLouisa  .
## age             0.0136619997
## gendermale      .
## height         -0.1105084258
## framemedium     0.0433562567
## framesmall      .
## bp.1s           0.0101793229
## time.ppn        0.0006126125
## bmi             0.0046024506
## whr             .
```

```
# Predict probabilities on the test set
lasso_prob <- predict(lasso_model, s = cv_lasso$lambda.min, newx = x_test, type = "response")

# Convert probabilities to binary predictions (0 or 1)
lasso_pred <- ifelse(lasso_prob > 0.5, 1, 0)

# Ensure predictions and actual values are integers for comparison
lasso_pred <- as.integer(lasso_pred)
y_test <- as.integer(y_test)

# Confusion Matrix
lasso_conf_matrix <- table(Predicted = lasso_pred, Actual = y_test)
cat("Confusion Matrix (Lasso):\n")
```

```
## Confusion Matrix (Lasso):
```

```
print(lasso_conf_matrix)
```

```
##          Actual
## Predicted  0  1
##         0 77  6
##         1  2  8
```

```
# Misclassification Rate
lasso_misclass_rate <- sum(lasso_pred != y_test)/length(y_test)
cat("Misclassification Rate (Lasso):", round(lasso_misclass_rate, 4), "\n")
```

```
## Misclassification Rate (Lasso): 0.086
```

A **sparse logistic regression model** is a variant of logistic regression that includes a **regularization term** to penalize complex models and encourage sparsity (i.e., setting some coefficients to zero).

Interpretation of Lasso Regression Results

1. **Optimal Lambda**:
   - The **optimal lambda** value selected by cross-validation is `0.01122084`. This value minimizes the **binomial deviance** and provides a balance between model complexity and prediction accuracy.
2. **Non-Zero Coefficients**:
   - Lambda controls the amount of regularization applied to the model. A larger value of results in more shrinkage of the coefficients toward zero.
   - ndefined**Lasso regression** has performed feature selection by shrinking some coefficients to zero. Out of the original predictors, the following remain in the model:
     - `chol` (positive effect)
     - `stab.glu` (strong positive effect)
     - `age` (positive effect)
     - `height` (negative effect)
     - `framemedium` (positive effect)
     - `bp.1s` (positive effect)
     - `time.ppn` (slightly positive effect)
     - `bmi` (slightly positive effect)
   - Predictors like `hdl`, `locationLouisa`, `gendermale`, `framesmall`, and `whr` were shrunk to zero, indicating they **do not contribute significantly** to the prediction of diabetes.
3. **Confusion Matrix**:
   - The model **correctly predicts 77 negatives** and **8 positives**.
   - There are **6 false negatives** and **2 false positives**, indicating the model performs better for the negative class but still struggles with detecting positives.
4. **Misclassification Rate**:
   - Despite achieving a low misclassification rate of 0.086, the model's ability to predict the positive class remains limited

3. Generalized Additive Model (GAM)

```
names(diabetes_data)
```

```
##  [1] "chol"     "stab.glu" "hdl"      "location" "age"      "gender"
##  [7] "height"   "frame"    "bp.1s"    "time.ppn" "bmi"      "dtest"
## [13] "whr"
```

```
# Load the necessary package
library(mgcv)

# Fit a GAM model (using only linear terms for all predictors)
gam_model <- gam(dtest ~ chol + stab.glu + age + bmi + location + gender + frame,
    data = train_data, family = "binomial")
# gam_model <- gam(dtest ~ chol + stab.glu + hdl + location + age + gender +
# height + frame + bp.1s + time.ppn + bmi + whr, data = train_data, family =
# 'binomial')

# Summary of the model
summary(gam_model)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## dtest ~ chol + stab.glu + age + bmi + location + gender + frame
##
## Parametric coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -13.505070   2.761255  -4.891 1.00e-06 ***
## chol             0.008642   0.006981   1.238   0.2157
## stab.glu         0.049101   0.008663   5.668 1.45e-08 ***
## age              0.045634   0.019804   2.304   0.0212 *
## bmi              0.048126   0.045840   1.050   0.2938
## locationLouisa   0.062409   0.588790   0.106   0.9156
## gendermale      -0.994232   0.745945  -1.333   0.1826
## framemedium      0.749905   0.741166   1.012   0.3116
## framesmall       0.440791   1.033421   0.427   0.6697
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.624   Deviance explained = 59.7%
## UBRE = -0.58723  Scale est. = 1         n = 282
```

```
# Predict on test data
gam_predictions <- predict(gam_model, newdata = test_data, type = "response")

# Convert probabilities to class labels
gam_class_predictions <- ifelse(gam_predictions > 0.5, 1, 0)

# Confusion matrix
gam_conf_matrix <- table(Predicted = gam_class_predictions, Actual = test_data$dtest)
print(gam_conf_matrix)
```

```
##          Actual
## Predicted  0  1
##         0 76  5
##         1  3  9
```

```
# Misclassification rate
gam_misclass_rate <- mean(gam_class_predictions != test_data$dtest)
cat("Misclassification Rate (GAM without Smoothness):", gam_misclass_rate, "\n")
```

```
## Misclassification Rate (GAM without Smoothness): 0.08602151
```

Generalized Additive Models (GAMs) allows for flexible, non-linear relationships between the predictors (independent variables) and the response (dependent variable). This flexibility is achieved using smooth functions to model these relationships.

Interpretation of the Results:

1. **Model Fit**:
   - The adjusted R^2 value is **0.624**, indicating that approximately 62.4% of the variation in the target variable (`dtest`) is explained by the predictors in the model.
   - The **Deviance explained** is **59.7%**, further supporting that the model fits the data reasonably well.
2. **Parametric Coefficients**:
   - **Intercept**: The intercept is highly significant ($p < 0.001$), representing the baseline log-odds of the positive class when all predictors are at their reference or mean values.
   - **Significant Predictors**:
     - **stab.glu** ($p < 0.001$): Highly significant and positively associated with diabetes risk. A one-unit increase in stabilized glucose increases the log-odds of diabetes by **0.049**.
     - **age** ($p = 0.0212$): Significant and positively associated with diabetes risk. Older individuals have a higher likelihood of being classified as diabetic.
   - **Non-Significant Predictors**:
     - Predictors such as **chol**, **bmi**, **locationLouisa**, **gender**, and **frame** have $p > 0.05$, suggesting they are not strong predictors of diabetes in this model.
3. **Classification Performance**:
   - **Confusion Matrix**:
     - True Negatives (Correctly classified non-diabetics): **76**
     - True Positives (Correctly classified diabetics): **9**
     - False Negatives (Missed diabetics): **5**
     - False Positives (Non-diabetics misclassified as diabetics): **3**
   - **Misclassification Rate**:
     - The model achieves a misclassification rate of **8.6%**, meaning about 91.4% of the test samples were correctly classified. This is a strong performance for a straightforward logistic regression model.
4. **Implications**:
   - **stab.glu** is the most important predictor of diabetes, followed by **age**, while the other predictors have limited impact.
   - The low misclassification rate and well-balanced confusion matrix indicate that the model is performing well, with minimal bias toward either class.
5. **Limitations**:
   - Some predictors with high p-values could potentially be removed to simplify the model without losing much predictive power.
   - The model could still benefit from incorporating interactions or non-linear terms for predictors like **chol** or **bmi** if their relationships with diabetes are complex.

a)

```
# Fit the GAM model
gam_model <- gam(dtest ~ s(chol) + s(stab.glu) + s(age) + s(bmi) + s(bp.1s) + location +
    gender + frame, data = train_data, family = "binomial")

# Summary of the GAM model
cat("GAM Model Summary:\n")
```

```
## GAM Model Summary:
```

```r
print(summary(gam_model))
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## dtest ~ s(chol) + s(stab.glu) + s(age) + s(bmi) + s(bp.1s) +
##     location + gender + frame
##
## Parametric coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -3.8651     0.9013  -4.288  1.8e-05 ***
## locationLouisa  -0.2907     0.6441  -0.451    0.652
## gendermale      -0.5378     0.7449  -0.722    0.470
## framemedium      1.1515     0.7996   1.440    0.150
## framesmall       1.2566     1.1924   1.054    0.292
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##               edf Ref.df Chi.sq p-value
## s(chol)     2.699  3.475  6.847   0.101
## s(stab.glu) 1.000  1.000 28.965  <2e-16 ***
## s(age)      1.000  1.000  0.243   0.622
## s(bmi)      4.944  6.054  9.013   0.177
## s(bp.1s)    1.588  1.977  3.340   0.161
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.712   Deviance explained = 69.6%
## UBRE = -0.62147  Scale est. = 1         n = 282
```

```r
# Predict on the test set
gam_predictions <- predict(gam_model, newdata = test_data, type = "response")

# Convert probabilities to class labels
gam_class_predictions <- ifelse(gam_predictions > 0.5, 1, 0)

# Confusion matrix
gam_conf_matrix <- table(Predicted = gam_class_predictions, Actual = test_data$dtest)
cat("Confusion Matrix (GAM):\n")
```

```
## Confusion Matrix (GAM):
```

```r
print(gam_conf_matrix)
```

```
##          Actual
## Predicted  0  1
##         0 74  7
##         1  5  7
```

```r
# Misclassification rate
gam_misclass_rate <- mean(gam_class_predictions != test_data$dtest)
cat("Misclassification Rate (GAM):", gam_misclass_rate, "\n")
```

```
## Misclassification Rate (GAM): 0.1290323
```

We fit a **Generalized Additive Model (GAM)** to predict the binary outcome (`dtest`) using a combination of continuous and categorical predictors. This allows us to model **non-linear relationships** for numeric variables while including categorical predictors as standard linear terms.

1. **Chosen Formula**:
   - We included **smooth functions** (`s(variable)`) for the **continuous predictors** to allow flexibility in modeling non-linear relationships:
     - `s(chol)` → Total cholesterol
     - `s(stab.glu)` → Stabilized glucose
     - `s(age)` → Age of the individual
     - `s(bmi)` → Body Mass Index
     - `s(bp.1s)` → Systolic blood pressure
   - We included **factor variables** (categorical predictors) without smoothing:
     - `location` → Location (e.g., Buckingham, Louisa)
     - `gender` → Male/Female
     - `frame` → Body frame size (large, medium, small)

   Final Formula:

$$\text{dtest} \sim s(\text{chol}) + s(\text{stab.glu}) + s(\text{age}) + s(\text{bmi}) + s(\text{bp.1s}) + \text{location} + \text{gender} + \text{frame}$$

Why Did We Choose This Formula?

1. **Continuous Predictors**:
   - Using `s()` on numeric predictors allows the model to flexibly adapt to **non-linear relationships** that might exist between these variables and the target (`dtest`).
2. **Factor Predictors**:
   - Categorical variables like `location`, `gender`, and `frame` are included **linearly** because smooth functions (`s()`) are not meaningful for non-numeric data.
3. **Balancing Complexity**:
   - By applying smoothing only where it makes sense, we reduce unnecessary complexity while still capturing important patterns in the data.

Interpretation of the GAM Results:

1. **Model Summary**:
   - The **GAM** model uses a **binomial family** with a logit link to predict the binary outcome `dtest` (presence/absence of diabetes).
   - The formula includes smooth terms for continuous variables (**chol**, **stab.glu**, **age**, **bmi**, **bp.1s**) and linear terms for categorical variables (**location**, **gender**, **frame**).
2. **Parametric Coefficients**:
   - **Intercept**: Statistically significant with a negative estimate (-3.8651), suggesting a lower baseline log-odds for the positive class.
   - **Location**, **Gender**, and **Frame**: All have **high p-values** (> 0.05), indicating that they are not statistically significant predictors of the outcome.
3. **Smooth Terms**:
   - **s(stab.glu)**: Highly significant (**p-value $<$ 2e-16**) and contributes meaningfully to the model, showing a strong non-linear relationship with the outcome.
   - **s(chol)**, **s(bmi)**, and **s(bp.1s)**: Not significant but suggest some non-linear trends (e.g., higher effective degrees of freedom, `edf`).
   - **s(age)**: Shows no significant effect, suggesting age may not play a strong role in this model.
4. **Model Performance**:
   - **Deviance Explained**: The model explains **69.6%** of the deviance, indicating a reasonably good fit.
   - **Adjusted R-sq**: Approximately **0.71**, showing a high proportion of variance explained by the model.
5. **Confusion Matrix and Misclassification Rate**:
   - The model correctly classified most observations but struggled with identifying positive cases (diabetes), as shown by the **7 false negatives**.
   - **Misclassification Rate**: **12.9%** (0.129), indicating that the model misclassified ~13% of the observations.

Smoothness should only be applied to variables with evidence of non-linear relationships. In our dataset, most relationships appear linear, making smoothness unnecessary. Simpler models often perform better on small datasets or when the relationships between variables and the target are straightforward.

b)

```
# Fit the GAM model with limited degrees of freedom for smooth terms
gam_model <- gam(dtest ~ s(chol, k = 5) + s(stab.glu, k = 5) + s(age, k = 4) + s(bmi,
    k = 5) + s(bp.1s, k = 5) + location + gender + frame, data = train_data, family = "binomial")

# Summary of the GAM model
cat("GAM Model Summary with Limited Degrees of Freedom:\n")
```

```
## GAM Model Summary with Limited Degrees of Freedom:
```

```
print(summary(gam_model))
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## dtest ~ s(chol, k = 5) + s(stab.glu, k = 5) + s(age, k = 4) +
##     s(bmi, k = 5) + s(bp.1s, k = 5) + location + gender + frame
##
## Parametric coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -3.8771     0.9157  -4.234  2.3e-05 ***
## locationLouisa  -0.4295     0.6448  -0.666    0.505
## gendermale      -0.6041     0.7220  -0.837    0.403
## framemedium      1.2071     0.8101   1.490    0.136
## framesmall       1.1140     1.1590   0.961    0.336
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##               edf Ref.df Chi.sq  p-value
## s(chol)     3.538  3.868  9.059   0.0553 .
## s(stab.glu) 1.000  1.000 26.075 1.63e-06 ***
## s(age)      1.000  1.000  0.501   0.4792
## s(bmi)      1.000  1.001  1.306   0.2535
## s(bp.1s)    1.907  2.381  3.980   0.1876
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.67   Deviance explained = 66.2%
## UBRE = -0.6117  Scale est. = 1         n = 282
```

```
# Predict on the test set
gam_predictions <- predict(gam_model, newdata = test_data, type = "response")

# Convert probabilities to class labels
gam_class_predictions <- ifelse(gam_predictions > 0.5, 1, 0)

# Confusion matrix
gam_conf_matrix <- table(Predicted = gam_class_predictions, Actual = test_data$dtest)
cat("Confusion Matrix (GAM with Limited Degrees of Freedom):\n")
```

```
## Confusion Matrix (GAM with Limited Degrees of Freedom):
```

```
print(gam_conf_matrix)
```

```
##          Actual
## Predicted  0  1
##         0 76  5
##         1  3  9
```

```
# Misclassification rate
gam_misclass_rate <- mean(gam_class_predictions != test_data$dtest)
```

```
cat("Misclassification Rate (GAM with Limited Degrees of Freedom):", gam_misclass_rate,
    "\n")
```

```
## Misclassification Rate (GAM with Limited Degrees of Freedom): 0.08602151
```

When the model draws a curve to show the relationship between a predictor (like age) and the response (like diabetes), it splits the curve into pieces to make it smooth. k tells the model how many pieces (or bends) it can use to draw the curve. A small k means the curve is simpler (fewer bends), while a large k allows the curve to wiggle more.

Solution: Limiting the Degrees of Freedom with `k` The `k` parameter in the `s()` function sets an **upper bound on the effective degrees of freedom** for the smooth term, effectively limiting the number of parameters that the model can estimate for that variable. Where: - `variable` is the numeric predictor being smoothed. - `k` is the maximum number of basis functions (related to the degrees of freedom). A smaller `k` limits the model's flexibility.

When to Use `k` - Small Sample Sizes: If your dataset is small, reducing `k` ensures the model does not overfit. - **Nearly Linear Relationships:** If the relationship between a variable and the target is almost linear, setting a small `k` encourages the model to use a simpler, less flexible smooth. - **Computational Efficiency:** Reducing `k` can speed up model fitting.

Interpretation of Results

- The **adjusted R-squared (R-sq.)** of 0.67 indicates that the model explains **67% of the variation** in the target variable, which is relatively strong.
- The **deviance explained (66.2%)** supports the conclusion that the model captures a significant portion of the variability in the data.
- The **UBRE score (-0.6117)** is a measure of model fit; lower values generally indicate better fit for GAMs.

Parametric Coefficients - The **intercept** is significant (**p-value $<$ 0.001**), indicating the model baseline prediction without considering predictors. - None of the categorical predictors (**location, gender, and frame**) are statistically significant, with **p-values $>$ 0.05**. - Example: `locationLouisa` has a p-value of 0.505, showing that the location does not contribute significantly to predicting diabetes status.

Comparison to GAM Without Limited Degrees of Freedom - The model with limited degrees of freedom achieves a similar misclassification rate (**8.6%**) as the model without limits. - However, by constraining the degrees of freedom (`k`), the model is less likely to overfit and is computationally more efficient.

1. **Stab.glu (stabilized glucose)** is the most influential predictor, with a highly significant smooth term (**p $<$ 0.001**).
2. **Chol (cholesterol)** shows a marginally significant non-linear effect, suggesting further investigation might improve the model.
3. Constraining the degrees of freedom through `k` helped reduce overfitting risks without compromising predictive accuracy.
4. The low misclassification rate demonstrates that the model effectively distinguishes between diabetic and non-diabetic cases, but further refinement could address false negatives.

c)

Significant Variables in the Model

1. **Parametric Terms:**

- None of the categorical variables (**location**, **gender**, **frame**) are significant:
  - Example: `locationLouisa` has a **p-value = 0.505**, indicating no significant contribution to the model.
- The intercept is statistically significant (**p < 0.001**), setting the baseline for predictions.

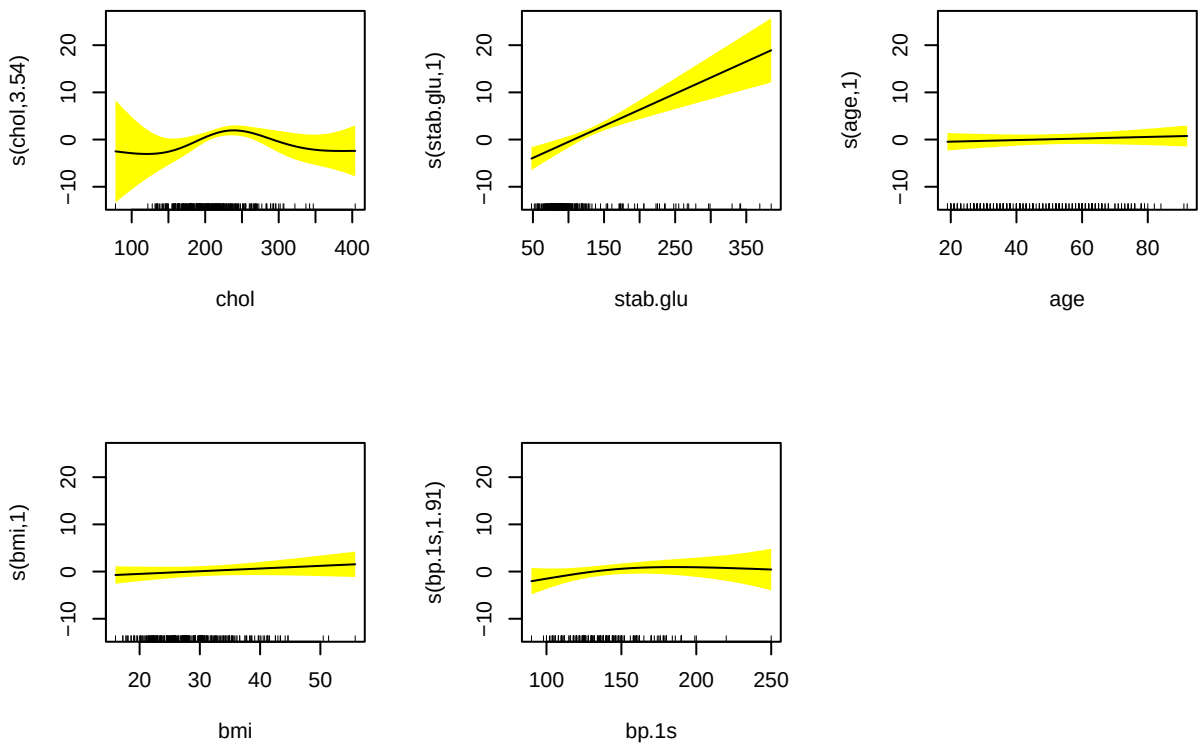
2. **Smooth Terms:**

- **s(stab.glu)** (stabilized glucose) is **highly significant** with **p < 0.001**, indicating a strong influence on diabetes prediction. This smooth function is essential for capturing the relationship between glucose levels and diabetes risk.
- **s(chol)** (cholesterol) is **marginally significant** with **p = 0.0553**, suggesting it may have a weak non-linear effect on the target variable.
- Other smooth terms:
  - **s(age)**, **s(bmi)**, and **s(bp.1s)** are not significant (**p > 0.05**), implying no strong evidence of non-linear relationships for these predictors.

Complexity of the Smooth Functions The complexity of smooth functions is determined by the **effective degrees of freedom (edf)**: - **s(chol)**: **edf = 3.538**, indicating moderate complexity. This suggests the function is flexible enough to capture non-linear patterns but is not overly complex. - **s(stab.glu)**: **edf = 1.000**, indicating a **linear effect**. This means the relationship is nearly linear and does not require additional flexibility. - **s(age)**, **s(bmi)**, and **s(bp.1s)**: - **edf = 1.000** for **age** and **bmi**, meaning these are treated as linear terms. - **s(bp.1s)** has **edf = 1.907**, indicating a slight non-linearity, but its contribution is not statistically significant (**p = 0.1876**).

d)

```r
# Plot smoothed terms with shaded confidence intervals
plot(gam_model, page = 1, shade = TRUE, shade.col = "yellow", seWithMean = TRUE)
```



What This Does The plot shows how the explanatory variables (**chol**, **stab.glu**, **age**, **bmi**, and **bp.1s**) relate to the response variable (**dtest**) in the model. Each curve represents the smoothed effect of the variable on the log-odds of the response. The shaded yellow areas show the confidence intervals, which indicate the level of certainty around the estimates. A flat line at zero means the variable has no effect.

Interpretation of the Plots

1. **s(chol):**
   The relationship between **chol** and the response is weak and slightly non-linear. The curve dips, flattens, and then rises, but the wide yellow confidence intervals at extreme values show there is high uncertainty for very low and very high **chol** levels. This suggests there are fewer observations in these ranges.

2. **s(stab.glu):**
   There is a clear positive trend between **stab.glu** and the response. As **stab.glu** increases, the probability of a positive response (dtest = 1) also increases. The narrow confidence intervals show strong certainty about this effect, making it the most important predictor.

3. **s(age):**
   The curve for **age** is almost flat, showing that it has little or no impact on predicting the response. The narrow confidence intervals confirm that this variable is not significant in the model.

4. **s(bmi):**
   The curve for **bmi** shows a slight upward trend, indicating a small positive association with the response. However, the effect is subtle and does not contribute much to the model.

5. **s(bp.1s):**
   The relationship for **bp.1s** is slightly non-linear, with a small dip followed by a rise. The effect is weak, and the confidence intervals widen at the extremes, showing less certainty for very low and very high values.

Key Takeaways

- **stab.glu** is the most important variable, with a clear and strong positive effect.
- Variables like **chol** and **bp.1s** show weak, non-linear effects, while **age** and **bmi** contribute very little to the model.
- Confidence intervals are wider at the extremes of some variables, indicating uncertainty due to fewer observations in these ranges.
- Overall, **stab.glu** stands out as the most reliable predictor, while other variables have limited or weak contributions.

e)

```
# Confusion matrix
gam_conf_matrix <- table(Predicted = gam_class_predictions, Actual = test_data$dtest)
cat("Confusion Matrix (GAM with Limited Degrees of Freedom):\n")
```

```
## Confusion Matrix (GAM with Limited Degrees of Freedom):
```

```
print(gam_conf_matrix)
```

```
##          Actual
## Predicted  0  1
##         0 76  5
##         1  3  9
```

```
# Misclassification rate
gam_misclass_rate <- mean(gam_class_predictions != test_data$dtest)
cat("Misclassification Rate (GAM with Limited Degrees of Freedom):", gam_misclass_rate,
    "\n")
```

```
## Misclassification Rate (GAM with Limited Degrees of Freedom): 0.08602151
```

f)

```
library(gam)
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.22-5

##
## Attaching package: 'gam'

## The following objects are masked from 'package:mgcv':
##
##     gam, gam.control, gam.fit, s
```

```
# ??gam help('gam', package = 'gam') conflicts()

# find('step.gam')

# sessionInfo()

# ls('package:gam')
```

```
# Start with all variables
current_formula <- dtest ~ s(chol) + s(stab.glu) + s(age) + s(bmi) + s(bp.1s) + location +
    gender + frame

# Fit the initial model
initial_gam <- gam(current_formula, data = train_data, family = "binomial")
cat("Initial Model Summary:\n")
```

```
## Initial Model Summary:
```

```
print(summary(initial_gam))
```

```
##
## Call: gam(formula = current_formula, family = "binomial", data = train_data)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.54054 -0.21234 -0.08079 -0.02182  2.71230
##
## (Dispersion Parameter for binomial family taken to be 1)
##
##     Null Deviance: 244.2261 on 281 degrees of freedom
## Residual Deviance: 61.3781 on 257.0001 degrees of freedom
## AIC: 111.3779
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##              Df Sum Sq Mean Sq  F value    Pr(>F)
## s(chol)       1  0.946   0.946   2.4368  0.119749
## s(stab.glu)   1 54.854  54.854 141.3175 < 2.2e-16 ***
## s(age)        1  0.225   0.225   0.5798  0.447096
## s(bmi)        1  0.653   0.653   1.6819  0.195833
## s(bp.1s)      1  2.836   2.836   7.3050  0.007334 **
## location      1  0.003   0.003   0.0071  0.932777
## gender        1  1.699   1.699   4.3775  0.037397 *
```

```
## frame         2  2.395   1.197   3.0847  0.047442 *
## Residuals   257 99.757   0.388
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##             Npar Df Npar Chisq  P(Chi)
## (Intercept)
## s(chol)           3    10.2253 0.01674 *
## s(stab.glu)       3     8.1092 0.04380 *
## s(age)            3     4.3159 0.22930
## s(bmi)            3     8.5384 0.03610 *
## s(bp.1s)          3     3.1324 0.37170
## location
## gender
## frame
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Step 1: Identify insignificant variables Look at p-values and smooth terms
# for manual decisions
insignificant_vars <- c("age", "bmi", "location", "gender", "frame")

# Step 2: Iteratively drop variables and refit
for (var in insignificant_vars) {
    cat("\nDropping:", var, "\n")

    # Update formula by removing the variable
    current_formula <- update(current_formula, paste(". ~ . -", var))

    # Refit the model
    updated_gam <- mgcv::gam(current_formula, data = train_data, family = "binomial")
    print(summary(updated_gam))

    # Predict on test data
    gam_pred <- predict(updated_gam, newdata = test_data, type = "response")
    gam_class <- ifelse(gam_pred > 0.5, 1, 0)

    # Evaluate performance
    conf_matrix <- table(Predicted = gam_class, Actual = test_data$dtest)
    cat("Confusion Matrix:\n")
    print(conf_matrix)

    misclass_rate <- mean(gam_class != test_data$dtest)
    cat("Misclassification Rate:", misclass_rate, "\n")

    # Decide to keep or drop the variable based on misclassification rate
    if (misclass_rate > 0.2) {
        cat(var, "seems important, re-adding it back.\n")
        current_formula <- update(current_formula, paste(". ~ . +", var))
    }
}
```

```
##
```

```
## Dropping: age
##
## Family: binomial
## Link function: logit
##
## Formula:
## dtest ~ s(chol) + s(stab.glu) + s(age) + s(bmi) + s(bp.1s) +
##     location + gender + frame
##
## Parametric coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -3.8651     0.9013  -4.288  1.8e-05 ***
## locationLouisa  -0.2907     0.6441  -0.451    0.652
## gendermale      -0.5378     0.7449  -0.722    0.470
## framemedium      1.1515     0.7996   1.440    0.150
## framesmall       1.2566     1.1924   1.054    0.292
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##               edf Ref.df Chi.sq p-value
## s(chol)     2.699  3.475  6.847   0.101
## s(stab.glu) 1.000  1.000 28.965  <2e-16 ***
## s(age)      1.000  1.000  0.243   0.622
## s(bmi)      4.944  6.054  9.013   0.177
## s(bp.1s)    1.588  1.977  3.340   0.161
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.712   Deviance explained = 69.6%
## UBRE = -0.62147  Scale est. = 1         n = 282
## Confusion Matrix:
##          Actual
## Predicted  0  1
##         0 74  7
##         1  5  7
## Misclassification Rate: 0.1290323
##
## Dropping: bmi
##
## Family: binomial
## Link function: logit
##
## Formula:
## dtest ~ s(chol) + s(stab.glu) + s(age) + s(bmi) + s(bp.1s) +
##     location + gender + frame
##
## Parametric coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -3.8651     0.9013  -4.288  1.8e-05 ***
## locationLouisa  -0.2907     0.6441  -0.451    0.652
## gendermale      -0.5378     0.7449  -0.722    0.470
## framemedium      1.1515     0.7996   1.440    0.150
## framesmall       1.2566     1.1924   1.054    0.292
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##               edf Ref.df Chi.sq p-value
## s(chol)     2.699  3.475  6.847   0.101
## s(stab.glu) 1.000  1.000 28.965  <2e-16 ***
## s(age)      1.000  1.000  0.243   0.622
## s(bmi)      4.944  6.054  9.013   0.177
## s(bp.1s)    1.588  1.977  3.340   0.161
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.712   Deviance explained = 69.6%
## UBRE = -0.62147  Scale est. = 1         n = 282
## Confusion Matrix:
##          Actual
## Predicted  0  1
##         0 74  7
##         1  5  7
## Misclassification Rate: 0.1290323
##
## Dropping: location
##
## Family: binomial
## Link function: logit
##
## Formula:
## dtest ~ s(chol) + s(stab.glu) + s(age) + s(bmi) + s(bp.1s) +
##     gender + frame
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.8705     0.8930  -4.334 1.46e-05 ***
## gendermale   -0.5631     0.7485  -0.752    0.452
## framemedium   1.0514     0.7648   1.375    0.169
## framesmall    1.1264     1.1650   0.967    0.334
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##               edf Ref.df Chi.sq p-value
## s(chol)     2.557  3.303  6.455   0.107
## s(stab.glu) 1.000  1.000 29.801  <2e-16 ***
## s(age)      1.000  1.000  0.301   0.583
## s(bmi)      4.900  6.009  8.986   0.175
## s(bp.1s)    1.613  2.013  3.630   0.164
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.71   Deviance explained = 69.3%
## UBRE = -0.62701  Scale est. = 1         n = 282
## Confusion Matrix:
##          Actual
```

```
## Predicted  0  1
##         0 74  7
##         1  5  7
## Misclassification Rate: 0.1290323
##
## Dropping: gender
##
## Family: binomial
## Link function: logit
##
## Formula:
## dtest ~ s(chol) + s(stab.glu) + s(age) + s(bmi) + s(bp.1s) +
##     frame
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -4.2190     0.7839  -5.382 7.36e-08 ***
## framemedium   1.1757     0.7476   1.573    0.116
## framesmall    1.3612     1.1012   1.236    0.216
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##               edf Ref.df Chi.sq p-value
## s(chol)     2.648  3.407  7.357  0.0777 .
## s(stab.glu) 1.000  1.000 30.475  <2e-16 ***
## s(age)      1.000  1.000  0.325  0.5689
## s(bmi)      4.931  6.045 10.349  0.1130
## s(bp.1s)    1.568  1.950  3.176  0.1622
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.704   Deviance explained = 69.2%
## UBRE = -0.6325  Scale est. = 1         n = 282
## Confusion Matrix:
##          Actual
## Predicted  0  1
##         0 74  7
##         1  5  7
## Misclassification Rate: 0.1290323
##
## Dropping: frame
##
## Family: binomial
## Link function: logit
##
## Formula:
## dtest ~ s(chol) + s(stab.glu) + s(age) + s(bmi) + s(bp.1s)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.4178     0.5194  -6.581 4.69e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Approximate significance of smooth terms:
##               edf Ref.df Chi.sq p-value
## s(chol)     3.065  3.885  9.816  0.0397 *
## s(stab.glu) 1.000  1.000 31.855  <2e-16 ***
## s(age)      2.413  3.048  2.036  0.5659
## s(bmi)      5.071  6.199  9.857  0.1435
## s(bp.1s)    1.000  1.000  3.111  0.0778 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.706   Deviance explained = 69.3%
## UBRE = -0.638  Scale est. = 1         n = 282
## Confusion Matrix:
##          Actual
## Predicted  0  1
##         0 75  6
##         1  4  8
## Misclassification Rate: 0.1075269
```

```
# Final Model
cat("\nFinal Model Summary:\n")
```

```
##
## Final Model Summary:
```

```
final_gam <- mgcv::gam(current_formula, data = train_data, family = "binomial")
print(summary(final_gam))
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## dtest ~ s(chol) + s(stab.glu) + s(age) + s(bmi) + s(bp.1s)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.4178     0.5194  -6.581 4.69e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##               edf Ref.df Chi.sq p-value
## s(chol)     3.065  3.885  9.816  0.0397 *
## s(stab.glu) 1.000  1.000 31.855  <2e-16 ***
## s(age)      2.413  3.048  2.036  0.5659
## s(bmi)      5.071  6.199  9.857  0.1435
## s(bp.1s)    1.000  1.000  3.111  0.0778 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.706   Deviance explained = 69.3%
## UBRE = -0.638  Scale est. = 1         n = 282
```

Comparison of Misclassification Rates: The **GAM with limited degrees of freedom** had a lower misclassification rate of 8.6% compared to the **manually selected GAM**, which had a misclassification rate of 10.75%. The GAM with limited degrees of freedom controlled for complexity by restricting the smoothness of the predictors (`k`). This likely reduced overfitting and helped the model generalize better to new data.

The manually selected GAM was created by keeping variables that showed some statistical significance, but it retained predictors like `s(age)` and `s(bmi)` that did not contribute strongly to the model. This may have added unnecessary noise and slightly worsened the predictive performance compared to the restricted model.

Both models identified **s(chol)** and **s(stab.glu)** as the most important predictors. These variables consistently showed strong significance and were critical to the predictive performance of the models. However, the manually selected GAM also kept weaker predictors like `s(age)` and `s(bmi)`, which increased the complexity without a clear improvement in accuracy.

In summary, the GAM with restricted degrees of freedom performed better, achieving a lower misclassification rate while maintaining a simpler and more efficient model. The manually selected GAM, while still effective, could have been improved by focusing only on the most significant predictors.

g)

```
# Fit a GAM with selected variables from (f)
selected_formula <- dtest ~ s(chol) + s(stab.glu) + s(age) + s(bmi) + s(bp.1s)

selected_gam <- mgcv::gam(selected_formula, data = train_data, family = "binomial")

# Summary of the updated model
cat("Summary of GAM with Selected Variables:\n")
```

```
## Summary of GAM with Selected Variables:
```

```
print(summary(selected_gam))
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## dtest ~ s(chol) + s(stab.glu) + s(age) + s(bmi) + s(bp.1s)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.4178     0.5194  -6.581 4.69e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##               edf Ref.df Chi.sq p-value
## s(chol)     3.065  3.885  9.816  0.0397 *
## s(stab.glu) 1.000  1.000 31.855  <2e-16 ***
## s(age)      2.413  3.048  2.036  0.5659
## s(bmi)      5.071  6.199  9.857  0.1435
## s(bp.1s)    1.000  1.000  3.111  0.0778 .
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.706   Deviance explained = 69.3%
## UBRE = -0.638  Scale est. = 1         n = 282
```

```
# Predict on the test set
selected_gam_pred <- predict(selected_gam, newdata = test_data, type = "response")
selected_gam_class <- ifelse(selected_gam_pred > 0.5, 1, 0)

# Confusion matrix
conf_matrix <- table(Predicted = selected_gam_class, Actual = test_data$dtest)
cat("Confusion Matrix:\n")
```

```
## Confusion Matrix:
```

```
print(conf_matrix)
```

```
##          Actual
## Predicted  0  1
##         0 75  6
##         1  4  8
```

```
# Misclassification rate
misclass_rate <- mean(selected_gam_class != test_data$dtest)
cat("Misclassification Rate (Selected GAM):", misclass_rate, "\n")
```

```
## Misclassification Rate (Selected GAM): 0.1075269
```

The selected GAM model uses `chol`, `stab.glu`, `age`, `bmi`, and `bp.1s` as predictors. Among these, `chol` and `stab.glu` are the most important variables, showing significant effects on predicting `dtest`, while `bp.1s` is borderline significant. `age` and `bmi` do not have strong evidence of being significant in this model. The model explains about 70.6% of the variability in the data and has a misclassification rate of 10.75%, which is slightly higher than the GAM with limited degrees of freedom (8.6%). However, this rate is reasonable and demonstrates that the model balances simplicity and prediction accuracy well.