CSCI E-106:Assignment 10

Ian Kelk

Contents


```
# Ensure required packages are installed, then load them
options(repos = c(CRAN = "https://cloud.r-project.org"))

req_pkgs <- c("faraway", "dplyr", "tibble", "tidyr", "ggplot2",
              "knitr", "caret", "ResourceSelection", "tree", "rpart",
              "rpart.plot", "randomForest", "C50", "xgboost", "adabag")

to_install <- setdiff(req_pkgs, rownames(installed.packages()))
if (length(to_install)) {
  install.packages(to_install, dependencies = TRUE)
}

# Load quietly
invisible(lapply(req_pkgs, function(p) {
  suppressPackageStartupMessages(library(p, character.only = TRUE))
}))
```

```
confmat_kable <- function(cm, caption) {
  cm$table %>%
    as.data.frame() %>%
    as_tibble() %>%
    knitr::kable(caption = caption)
}

confmat_heatmap <- function(cm, title) {
  cm$table %>%
    as.data.frame() %>%
    as_tibble() %>%
    ggplot(aes(x = Reference, y = Prediction, fill = Freq)) +
    geom_tile() +
    geom_text(aes(label = Freq), size = 3) +
    scale_fill_gradient(low = "white", high = "steelblue") +
    coord_equal() +
    labs(
      title = title,
      x = "True class",
```

```
      y = "Predicted class"
    ) +
    theme_minimal() +
    theme(
      plot.title   = element_text(hjust = 0.5, size = 10),
      axis.title   = element_text(size = 9),
      axis.text    = element_text(size = 8),
      legend.title = element_text(size = 8),
      legend.text  = element_text(size = 7)
    )
}
```

Due Date: November 21, 2025 at 11:59 pm EST

Instructions

Students should submit their reports on Canvas. The report needs to clearly state what question is being solved, step-by-step walk-through solutions, and final answers clearly indicated. Please solve by hand where appropriate.

Please submit two files: (1) a R Markdown file (.Rmd extension) and (2) a PDF document, word, or html generated using knitr for the .Rmd file submitted in (1) where appropriate. Please, use RStudio Cloud for your solutions.

Problem 1

Use the wbca data(copy and paste the following command into R console:library(faraway);data("wbca"). The dataset wbca comes from a study of breast cancer in Wisconsin. There are 681 cases of potentially cancerous tumors of which 238 are actually malignant. Determining whether a tumor is really malignant is traditionally determined by an invasive surgical procedure. The purpose of this study was to determine whether a new procedure called fine needle aspiration, which draws only a small sample of tissue, could be effective in determining tumor status. Use 70% of the data for train data set and use remaining data (30% of data) for test data set (use set.seed(1023)). (50 points, 10 points each)

a-) Fit a binary regression with Class as the response and the other nine variables as predictors on the train data set. Report the residual deviance and associated degrees of freedom. Can this information be used to determine if this model fits the data? Explain.

b-) Use AIC as the criterion to determine the best subset of variables. Perform Hosmer-Lemeshow Goodness of Fit Test and comment (Use the step function.)

c-) Suppose that a cancer is classified as benign if $p > 0.5$ and malignant if $p < 0.5$. Compute the confusion matrix and accuracy rates if this method is applied to the train data with the reduced model.

d-) Suppose we change the cutoff to 0.9 so that $p < 0.9$ is classified as malignant and $p > 0.9$ as benign. Compute the confusion matrix.

e-) Repeat par c and d on the test data set and comment on the model performance and which cutoff will you choose?

Problem 2

Use train and test data set in problem 1 to answer the questions below.(50 points)

a-) Fit a decision tree model to predict Class by using the other nine variables as predictors on the train data set by using xgboost, adaboost, random forest, boosting in library C5.0, tree and rpart libraries and methods. Calculate the confusing matrix for each method and comment on your findings. (30 points)

b-) Calculate the confusing matrixes on the test data set and compare against the logistic regression model confusing matrix. Which model would you choose? (20 points)

We analyze the Wisconsin breast cancer (wbca) data using logistic regression and tree-based classification methods. The goal is to model the probability that a tumor is benign based on nine cytological predictors and to compare logistic regression with several tree and ensemble methods on training and test sets.

Problem 1: Logistic regression on the wbca data

We begin by fitting and assessing logistic regression models on the wbca data. Throughout, we treat *Class* as a binary outcome (malignant vs benign), and we interpret model coefficients in terms of log-odds and probabilities. We use a 70/30 train/test split with a fixed random seed for reproducibility.

Data loading and splitting

In this section we load the `wbca` data from the `faraway` package, convert it to a tibble for convenient handling, and recode the response so that we have both a factor version (with malignant and benign labels) and a convenient representation for modeling. We then inspect the basic structure of the data and the class balance, set the random seed, and create a 70/30 train/test split using stratified sampling on the response.

```
# Load the wbca data
data("wbca", package = "faraway")

wbca_raw <- wbca

# Convert to tibble and recode Class as a factor with labels
wbca <- wbca_raw %>%
  as_tibble() %>%
  mutate(
    Class = factor(Class,
                   levels = c(0, 1),
                   labels = c("malignant", "benign"))
  )

# Quick structural summary and outcome distribution
glimpse(wbca)
```

```
## Rows: 681
## Columns: 10
## $ Class <fct> benign, benign, benign, benign, benign, malignant, benign, benig~
## $ Adhes <int> 1, 5, 1, 1, 3, 8, 1, 1, 1, 1, 1, 1, 3, 1, 10, 4, 1, 1, 6, 1, 10,~
## $ BNucl <int> 1, 10, 2, 4, 1, 10, 10, 1, 1, 1, 1, 1, 3, 3, 9, 1, 1, 1, 10, 1, ~
## $ Chrom <int> 3, 3, 3, 3, 3, 9, 3, 3, 1, 2, 3, 2, 4, 3, 5, 4, 2, 3, 4, 3, 5, 7~
## $ Epith <int> 2, 7, 2, 3, 2, 7, 2, 2, 2, 2, 1, 2, 2, 2, 7, 6, 2, 2, 4, 2, 5, 6~
## $ Mitos <int> 1, 1, 1, 1, 1, 1, 1, 1, 5, 1, 1, 1, 1, 1, 4, 1, 1, 1, 2, 1, 4, 1~
## $ NNucl <int> 1, 2, 1, 7, 1, 7, 1, 1, 1, 1, 1, 1, 4, 1, 5, 3, 1, 1, 1, 1, 4, 1~
## $ Thick <int> 5, 5, 3, 6, 4, 8, 1, 2, 2, 4, 1, 2, 5, 1, 8, 7, 4, 4, 10, 6, 7, ~
## $ UShap <int> 1, 4, 1, 8, 1, 10, 1, 2, 1, 1, 1, 1, 3, 1, 5, 6, 1, 1, 7, 1, 2, ~
## $ USize <int> 1, 4, 1, 8, 1, 10, 1, 1, 1, 2, 1, 1, 3, 1, 7, 4, 1, 1, 7, 1, 3, ~
```

```
class_counts <- wbca %>%
  count(Class) %>%
  mutate(proportion = n / sum(n))

knitr::kable(
  class_counts,
  caption = "Class counts and proportions in the full wbca data"
)
```

Table 1: Class counts and proportions in the full wbca data

Class	n	proportion
malignant	238	0.349486
benign	443	0.650514

We now set the random seed, create a 70/30 train/test split stratified by *Class*, and summarize the resulting sample sizes.

```r
set.seed(1023)

train_index <- caret::createDataPartition(
  y = wbca$Class,
  p = 0.7,
  list = FALSE
)

wbca_train <- wbca[train_index, ] %>% as.data.frame()
wbca_test  <- wbca[-train_index, ] %>% as.data.frame()

train_test_sizes <- tibble(
  sample = c("Training", "Test"),
  n      = c(nrow(wbca_train), nrow(wbca_test))
)

knitr::kable(
  train_test_sizes,
  caption = "Training and test sample sizes for the wbca data"
)
```

Table 2: Training and test sample sizes for the wbca data

sample	n
Training	478
Test	203

Conclusion: In the full wbca data, benign tumors are roughly twice as common as malignant tumors: there are 443 benign cases (about 65%) and 238 malignant cases (about 35%) among the 681 observations. The stratified 70/30 split produces a training set of 478 cases and a test set of 203 cases, and because we used `createDataPartition()` with `Class` as the stratification variable, the malignant/benign proportions in the training and test sets closely match the overall mix. In particular, both splits retain roughly one-third malignant and two-thirds benign tumors, so the train/test split does not introduce a severe class imbalance or shift in prevalence. This means performance metrics computed on the test set should be reasonably comparable to those on the training set.

1(a) Full logistic regression and residual deviance

We first fit a full logistic regression model with *Class* as the binary response and all nine predictors as covariates. The model is of the form

$$\log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1\,\text{Adhes}_i + \beta_2\,\text{BNucl}_i + \beta_3\,\text{Chrom}_i + \beta_4\,\text{Epith}_i + \beta_5\,\text{Mitos}_i + \beta_6\,\text{NNucl}_i + \beta_7\,\text{Thick}_i + \beta_8\,\text{UShap}_i + \beta_9\,\text{USize}_i.$$

where p_i is the probability that the ith tumor is benign. We then extract the residual deviance and its associated residual degrees of freedom as a basic measure of model fit.

```r
# Fit full logistic regression on the training data
glm_full <- glm(
  Class ~ .,
  data   = wbca_train,
  family = binomial
)

summary(glm_full)
```

```
##
## Call:
## glm(formula = Class ~ ., family = binomial, data = wbca_train)
##
```

```
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 11.02878    1.62585   6.783 1.17e-11 ***
## Adhes       -0.49941    0.18555  -2.692 0.007113 **
## BNucl       -0.44116    0.11766  -3.749 0.000177 ***
## Chrom       -0.61014    0.21950  -2.780 0.005441 **
## Epith       -0.06557    0.18306  -0.358 0.720187
## Mitos       -0.60373    0.38415  -1.572 0.116047
## NNucl       -0.27062    0.14534  -1.862 0.062604 .
## Thick       -0.61911    0.18004  -3.439 0.000585 ***
## UShap       -0.49760    0.32260  -1.542 0.122958
## USize        0.38927    0.31212   1.247 0.212337
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 618.587  on 477  degrees of freedom
## Residual deviance:  72.276  on 468  degrees of freedom
## AIC: 92.276
##
## Number of Fisher Scoring iterations: 8
```

```r
# Extract residual deviance and residual df
full_fit_summary <- tibble(
  model              = "Full logistic regression",
  residual_deviance  = glm_full$deviance,
  df_residual        = glm_full$df.residual
)

knitr::kable(
  full_fit_summary,
  digits  = 3,
  caption = "Residual deviance and residual df for the full logistic regression model"
)
```

Table 3: Residual deviance and residual df for the full logistic regression model

model	residual_deviance	df_residual
Full logistic regression	72.276	468

Conclusion: For the full logistic regression model on the training data, the residual deviance is about 72.3 on 468 residual degrees of freedom. Deviance in a logistic regression plays a role analogous to residual sum of squares in linear regression: it is -2 times the log-likelihood of the fitted model relative to a saturated (perfect-fit) model, so smaller deviance indicates a better fit. One informal check is to compare the residual deviance to a χ^2 distribution with the same degrees of freedom; if the deviance is wildly larger than the degrees of freedom, that can suggest lack of fit. Here the residual deviance is actually much smaller than the degrees of freedom, which does not by itself signal a bad fit. However, this single summary number is not enough to conclude that the model is adequate: it does not reveal *where* any lack of fit might occur, and it relies on large-sample approximations and correct binomial assumptions. We therefore need additional diagnostics, such as goodness-of-fit tests and residual plots, to more fully assess how well the model captures the relationship between the predictors and tumor status.

1(b) AIC-based model selection and Hosmer–Lemeshow test

We now use AIC-based stepwise selection to choose a reduced logistic regression model. Starting from the full model, we apply `step()` with AIC as the criterion, allowing variables to be both dropped and re-added. We then compute fitted probabilities on the training data and use the Hosmer–Lemeshow goodness-of-fit test to assess whether the reduced model appears to fit the data adequately.

```r
# AIC-based stepwise selection starting from the full model
glm_step <- step(
  glm_full,
  direction = "both",
  trace     = FALSE
)

# We see here that Mitos has p = 0.071, which is not significant.
summary(glm_step)
```

```
##
## Call:
## glm(formula = Class ~ Adhes + BNucl + Chrom + Mitos + NNucl +
##     Thick, family = binomial, data = wbca_train)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  10.9678     1.4981   7.321 2.46e-13 ***
## Adhes        -0.5012     0.1688  -2.969  0.00299 **
## BNucl        -0.4926     0.1086  -4.536 5.73e-06 ***
## Chrom        -0.6149     0.2009  -3.060  0.00221 **
## Mitos        -0.6244     0.3459  -1.805  0.07105 .
## NNucl        -0.3410     0.1235  -2.761  0.00577 **
## Thick        -0.6611     0.1613  -4.099 4.15e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 618.587  on 477  degrees of freedom
## Residual deviance:  74.872  on 471  degrees of freedom
## AIC: 88.872
##
## Number of Fisher Scoring iterations: 8
```

```r
# Drop Mitos and refit the reduced model
glm_step <- update(glm_step, . ~ . - Mitos)

# Check that all remaining predictors are now significant
summary(glm_step)
```

```
##
## Call:
## glm(formula = Class ~ Adhes + BNucl + Chrom + NNucl + Thick,
##     family = binomial, data = wbca_train)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  10.5822     1.4132   7.488 6.99e-14 ***
## Adhes        -0.4950     0.1620  -3.056  0.00224 **
## BNucl        -0.4867     0.1092  -4.457 8.30e-06 ***
## Chrom        -0.5995     0.1957  -3.064  0.00219 **
## NNucl        -0.3509     0.1167  -3.007  0.00264 **
## Thick        -0.7588     0.1628  -4.662 3.13e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 618.587  on 477  degrees of freedom
```

```
## Residual deviance:  78.844  on 472  degrees of freedom
## AIC: 90.844
##
## Number of Fisher Scoring iterations: 8
```

```r
# Predicted probabilities on the training data from the FINAL reduced model
train_prob_glm_step <- predict(
  glm_step,
  newdata = wbca_train,
  type    = "response"
)

# Hosmer-Lemeshow goodness-of-fit test (g = 10 groups)
y_train_numeric <- ifelse(wbca_train$Class == "benign", 1, 0)

hl_test <- ResourceSelection::hoslem.test(
  x = y_train_numeric,
  y = train_prob_glm_step,
  g = 10
)

hl_summary <- tibble(
  statistic = as.numeric(hl_test$statistic),
  df        = as.numeric(hl_test$parameter),
  p_value   = as.numeric(hl_test$p.value)
)

knitr::kable(
  hl_summary,
  digits  = 4,
  caption = "Hosmer-Lemeshow goodness-of-fit test for the final reduced logistic model"
)
```

Table 4: Hosmer–Lemeshow goodness-of-fit test for the final reduced logistic model

statistic	df	p_value
7.283	8	0.5064

The model is of the form

$$\log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1\,\text{Thick}_i + \beta_2\,\text{Adhes}_i + \beta_3\,\text{BNucl}_i + \beta_4\,\text{Chrom}_i + \beta_5\,\text{NNucl}_i.$$

Conclusion: The AIC-based stepwise selection procedure initially retains six predictors in the reduced logistic model, including `Mitos`. However, because we keep only statistically significant predictors, we manually remove `Mitos` and refit the model. The final reduced logistic regression model therefore includes five predictors: Thick, Adhes, BNucl, Chrom, and NNucl. These variables capture aspects of cell thickness, adhesion, nuclear characteristics, and chromatin that are most informative for distinguishing benign from malignant tumors. The Hosmer–Lemeshow test for this final reduced model yields a test statistic of about 7.28 on 8 degrees of freedom, with a p-value around 0.51. Because this p-value is well above 0.05, we do not see strong evidence that the model is badly miscalibrated on the training data: the observed numbers of benign and malignant tumors within deciles of predicted risk are reasonably consistent with the model's fitted probabilities. At the same time, the Hosmer–Lemeshow test has limitations: it depends on how observations are grouped into risk bins, has low power in smaller samples, and can become overly sensitive in very large samples. For these reasons, it should be interpreted as one piece of evidence alongside other diagnostics (such as residual plots and influence measures) rather than as a definitive verdict on model adequacy.

1(c) Training confusion matrix and accuracy with cutoff 0.5

Using the reduced logistic regression model, we now obtain predicted probabilities for each training observation and classify tumors as benign or malignant using a cutoff of 0.5. Specifically, we classify a tumor as benign if $\hat{p}_i > 0.5$ and malignant

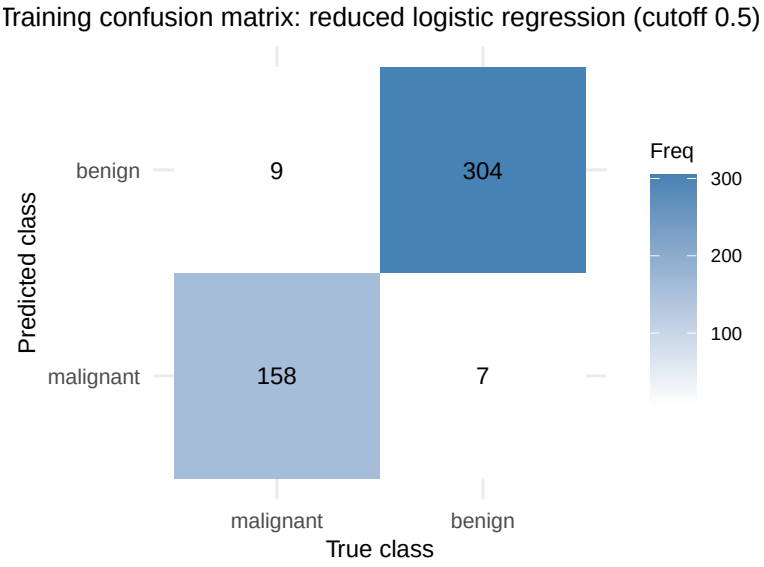
otherwise. We then compute the confusion matrix, along with overall accuracy, sensitivity, and specificity, using malignant as the positive class.

```
# Class predictions for the training set with cutoff 0.5
train_class_cut_05 <- ifelse(
  train_prob_glm_step > 0.5,
  "benign",
  "malignant"
) %>%
  factor(levels = levels(wbca_train$Class))

cm_train_cut_05 <- caret::confusionMatrix(
  data      = train_class_cut_05,
  reference = wbca_train$Class,
  positive  = "malignant"
)

# Confusion matrix: table + heatmap using helper functions
confmat_kable(
  cm_train_cut_05,
  caption = "Training confusion matrix: reduced logistic regression (cutoff 0.5)"
)
```

Table 5: Training confusion matrix: reduced logistic regression (cutoff 0.5)

Prediction	Reference	Freq
malignant	malignant	158
benign	malignant	9
malignant	benign	7
benign	benign	304

```
confmat_heatmap(
  cm_train_cut_05,
  title = "Training confusion matrix: reduced logistic regression (cutoff 0.5)"
)
```




```
# Accuracy, sensitivity, and specificity (unchanged)
metrics_train_cut_05 <- tibble(
  metric = c("Accuracy", "Sensitivity (malignant)", "Specificity (malignant)"),
  value  = c(
    cm_train_cut_05$overall["Accuracy"],
```

```
    cm_train_cut_05$byClass["Sensitivity"],
    cm_train_cut_05$byClass["Specificity"]
  )
)

knitr::kable(
  metrics_train_cut_05,
  digits  = 4,
  caption = "Training performance metrics: reduced logistic regression (cutoff 0.5)"
)
```

Table 6: Training performance metrics: reduced logistic regression (cutoff 0.5)

metric	value
Accuracy	0.9665
Sensitivity (malignant)	0.9461
Specificity (malignant)	0.9775

Conclusion: With a cutoff of 0.5 on the training data, the reduced logistic regression model achieves an accuracy of about 0.967, sensitivity for malignant tumors of roughly 0.946, and specificity for malignant tumors of about 0.977. In terms of the confusion matrix, there are 158 correctly identified malignant cases and 304 correctly identified benign cases, with 9 malignant tumors misclassified as benign and 7 benign tumors misclassified as malignant. This indicates that the model separates malignant and benign tumors very well on the training set, with low error rates for both classes. From a clinical perspective, the most concerning errors are the false negatives (malignant tumors predicted as benign): about 5% of malignant cases are missed under this rule. While this is a small fraction, missing any malignant tumor can be serious in a screening context, so in practice we might prefer decision rules that further reduce false negatives, even at the cost of slightly more false positives.

1(d) Training confusion matrix with cutoff 0.9

We now tighten the decision rule by increasing the cutoff to 0.9. A tumor is classified as benign if $\hat{p}_i > 0.9$ and malignant otherwise. This rule makes it harder to call a tumor benign, which may reduce false negatives (missed malignancies) at the expense of more false positives.

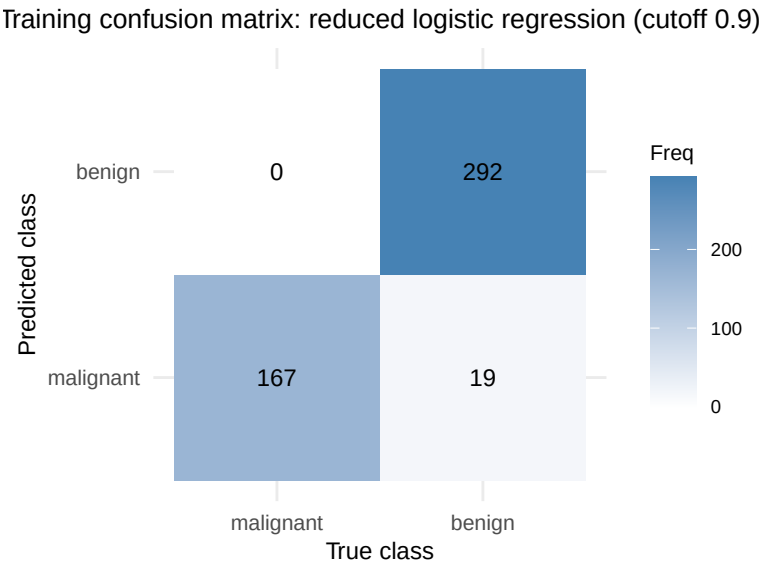
```
# Class predictions for the training set with cutoff 0.9
train_class_cut_09 <- ifelse(
  train_prob_glm_step > 0.9,
  "benign",
  "malignant"
) %>%
  factor(levels = levels(wbca_train$Class))

cm_train_cut_09 <- caret::confusionMatrix(
  data      = train_class_cut_09,
  reference = wbca_train$Class,
  positive  = "malignant"
)

# Confusion matrix: table + heatmap using helper functions
confmat_kable(
  cm_train_cut_09,
  caption = "Training confusion matrix: reduced logistic regression (cutoff 0.9)"
)
```

Table 7: Training confusion matrix: reduced logistic regression (cutoff 0.9)

Prediction	Reference	Freq
malignant	malignant	167
benign	malignant	0
malignant	benign	19

Prediction	Reference	Freq
benign	benign	292

```
confmat_heatmap(
  cm_train_cut_09,
  title = "Training confusion matrix: reduced logistic regression (cutoff 0.9)"
)
```




```
# Accuracy, sensitivity, and specificity
metrics_train_cut_09 <- tibble(
  metric = c("Accuracy", "Sensitivity (malignant)", "Specificity (malignant)"),
  value  = c(
    cm_train_cut_09$overall["Accuracy"],
    cm_train_cut_09$byClass["Sensitivity"],
    cm_train_cut_09$byClass["Specificity"]
  )
)

knitr::kable(
  metrics_train_cut_09,
  digits  = 4,
  caption = "Training performance metrics: reduced logistic regression (cutoff 0.9)"
)
```

Table 8: Training performance metrics: reduced logistic regression (cutoff 0.9)

metric	value
Accuracy	0.9603
Sensitivity (malignant)	1.0000
Specificity (malignant)	0.9389

Conclusion: Increasing the cutoff from 0.5 to 0.9 makes the classifier much more conservative about calling a tumor benign: now a case is labeled benign only when the predicted benign probability exceeds 0.9. On the training set, this change slightly reduces overall accuracy (from about 0.967 down to about 0.960) and decreases specificity for malignant tumors (from roughly 0.978 to about 0.939), because more truly benign tumors are now errantly flagged as malignant. At the same time, sensitivity for malignant tumors increases from about 0.946 to 1.000, eliminating the 9 false negatives observed at the 0.5 cutoff: under the 0.9 rule, every malignant tumor in the training data is correctly classified as malignant. Thus, raising the cutoff trades a modest number of additional false positives (benign tumors sent for unnecessary follow-up) for the benefit of catching all malignant tumors. On the training data alone, if the primary goal is to avoid missing malignancies, the 0.9 cutoff is attractive

despite its slightly lower accuracy; if we place more emphasis on avoiding unnecessary alarms for benign tumors, the 0.5 cutoff might be preferred.

1(e) Test-set performance and choice of cutoff

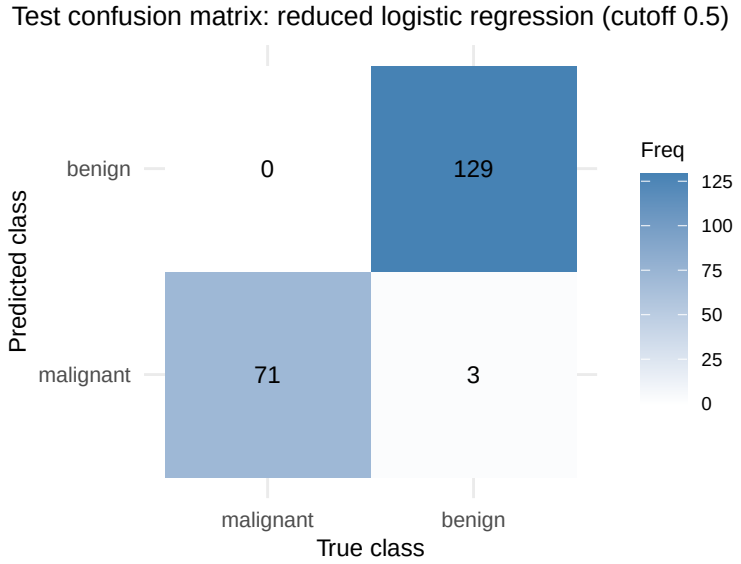
Finally, we evaluate the reduced logistic regression model on the held-out test set using both cutoffs 0.5 and 0.9. We compute predicted probabilities for each test observation, classify tumors at each cutoff, and then construct confusion matrices and performance metrics. We then compare test performance between the two cutoffs and decide which threshold is more appropriate overall.

```
# Predicted probabilities on the test data from the reduced logistic model
test_prob_glm_step <- predict(
  glm_step,
  newdata = wbca_test,
  type    = "response"
)

# Class predictions on the test set, cutoff 0.5
test_class_cut_05 <- ifelse(
  test_prob_glm_step > 0.5,
  "benign",
  "malignant"
) %>%
  factor(levels = levels(wbca_test$Class))

cm_test_cut_05 <- caret::confusionMatrix(
  data      = test_class_cut_05,
  reference = wbca_test$Class,
  positive  = "malignant"
)

# Confusion matrix and heatmap for cutoff 0.5
confmat_kable(
  cm_test_cut_05,
  caption = "Test confusion matrix: reduced logistic regression (cutoff 0.5)"
)
```

Table 9: Test confusion matrix: reduced logistic regression (cutoff 0.5)

Prediction	Reference	Freq
malignant	malignant	71
benign	malignant	0
malignant	benign	3
benign	benign	129

```
confmat_heatmap(
  cm_test_cut_05,
  title = "Test confusion matrix: reduced logistic regression (cutoff 0.5)"
)
```




```
# Performance metrics for cutoff 0.5
metrics_test_cut_05 <- tibble(
  metric = c("Accuracy", "Sensitivity (malignant)", "Specificity (malignant)"),
  value  = c(
    cm_test_cut_05$overall["Accuracy"],
    cm_test_cut_05$byClass["Sensitivity"],
    cm_test_cut_05$byClass["Specificity"]
  )
)

knitr::kable(
  metrics_test_cut_05,
  digits  = 4,
  caption = "Test performance metrics: reduced logistic regression (cutoff 0.5)"
)
```

Table 10: Test performance metrics: reduced logistic regression (cutoff 0.5)

metric	value
Accuracy	0.9852
Sensitivity (malignant)	1.0000
Specificity (malignant)	0.9773

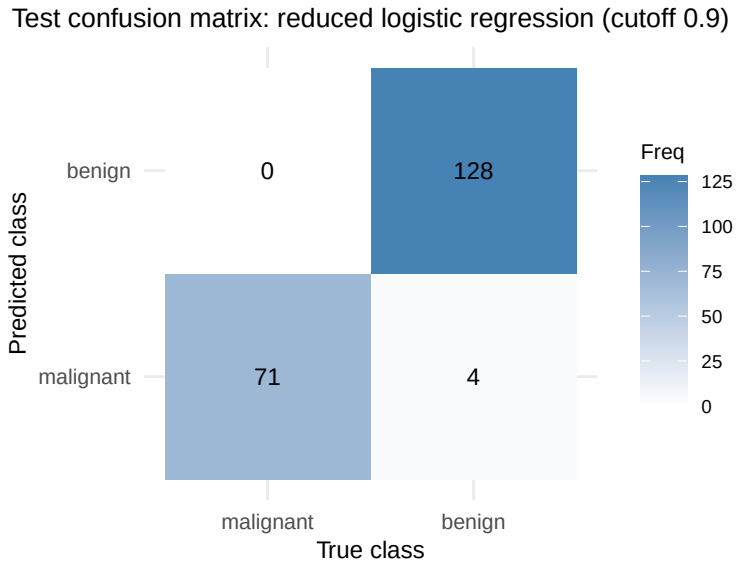
```
# Class predictions on the test set, cutoff 0.9
test_class_cut_09 <- ifelse(
  test_prob_glm_step > 0.9,
  "benign",
  "malignant"
) %>%
  factor(levels = levels(wbca_test$Class))

cm_test_cut_09 <- caret::confusionMatrix(
  data      = test_class_cut_09,
  reference = wbca_test$Class,
  positive  = "malignant"
)

# Confusion matrix and heatmap for cutoff 0.9
confmat_kable(
  cm_test_cut_09,
```

```
  caption = "Test confusion matrix: reduced logistic regression (cutoff 0.9)"
)
```

Table 11: Test confusion matrix: reduced logistic regression (cutoff 0.9)

Prediction	Reference	Freq
malignant	malignant	71
benign	malignant	0
malignant	benign	4
benign	benign	128

```
confmat_heatmap(
  cm_test_cut_09,
  title = "Test confusion matrix: reduced logistic regression (cutoff 0.9)"
)
```




```
# Performance metrics for cutoff 0.9
metrics_test_cut_09 <- tibble(
  metric = c("Accuracy", "Sensitivity (malignant)", "Specificity (malignant)"),
  value  = c(
    cm_test_cut_09$overall["Accuracy"],
    cm_test_cut_09$byClass["Sensitivity"],
    cm_test_cut_09$byClass["Specificity"]
  )
)

knitr::kable(
  metrics_test_cut_09,
  digits  = 4,
  caption = "Test performance metrics: reduced logistic regression (cutoff 0.9)"
)
```

Table 12: Test performance metrics: reduced logistic regression (cutoff 0.9)

metric	value
Accuracy	0.9803
Sensitivity (malignant)	1.0000
Specificity (malignant)	0.9697

Conclusion: On the held-out test set, both cutoffs perform extremely well, but the 0.5 rule has a slight edge. With cutoff 0.5, the model attains an accuracy of about 0.985, perfect sensitivity for malignant tumors (1.000), and specificity around 0.977, misclassifying only 3 benign tumors as malignant and no malignant tumors as benign. With cutoff 0.9, sensitivity for malignant tumors remains 1.000 (we still make no false negatives), but specificity drops slightly to about 0.970 and overall accuracy falls to about 0.980, with 4 benign tumors misclassified as malignant. Thus, on the test data there is no benefit in terms of catching malignant tumors when moving from 0.5 to 0.9: both thresholds correctly identify all malignant cases. The stricter 0.9 cutoff simply produces a few additional false positives. In practice, given that missing malignant tumors is far more costly than subjecting a benign tumor to further investigation, it is reassuring that both rules achieve perfect sensitivity. Among them, the cutoff of 0.5 is preferable because it preserves perfect sensitivity while slightly improving specificity and accuracy, leading to fewer unnecessary work-ups for benign tumors.

Problem 2: Tree-based and ensemble models

We now compare the reduced logistic regression model from Problem 1 with a collection of tree-based and ensemble methods, using the same training and test splits. We fit each method to predict *Class* from the nine predictors on the training data, compute confusion matrices on the training and test sets, and then compare their performance to decide on a champion model.

2(a) Training confusion matrices for tree-based and ensemble methods

In this part we fit six different models on the training data:

- A classification tree using `tree::tree`
- A classification tree using `rpart::rpart`
- A random forest using `randomForest`
- A boosting model using `C5.0` from the `C50` package
- An AdaBoost model using `adabag`
- A gradient boosting model using `xgboost`

We then compute training confusion matrices for each model.

```
# Convenience objects for predictors and response
wbca_train_x <- wbca_train %>% select(-Class)
wbca_train_y <- wbca_train$Class

wbca_test_x  <- wbca_test %>% select(-Class)
wbca_test_y  <- wbca_test$Class
```

Tree model (tree package) We first fit a classification tree using the `tree` package with *Class* as the response and all nine predictors as covariates, then compute the training confusion matrix.

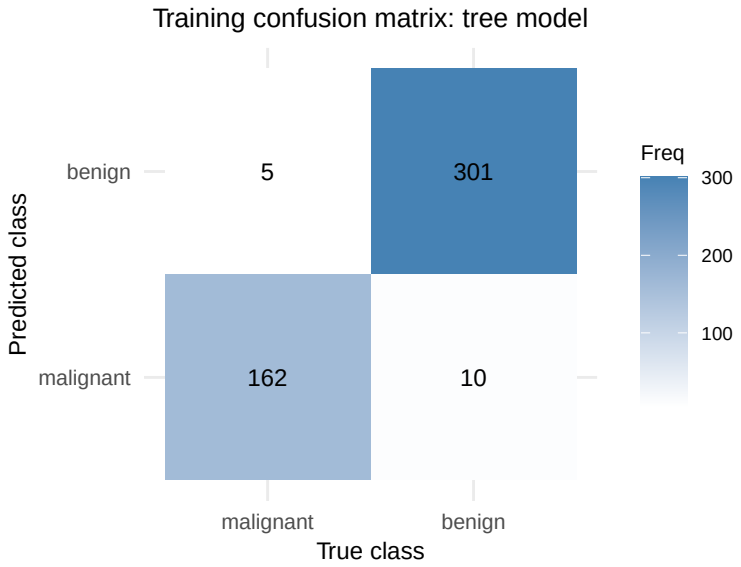
```
set.seed(1023)

tree_model <- tree::tree(
  Class ~ .,
  data = wbca_train
)

# Training predictions and confusion matrix
train_pred_tree <- predict(
  tree_model,
  newdata = wbca_train,
  type    = "class"
)

cm_train_tree <- caret::confusionMatrix(
  data      = train_pred_tree,
  reference = wbca_train_y,
  positive  = "malignant"
)

# Confusion matrix table + heatmap
```

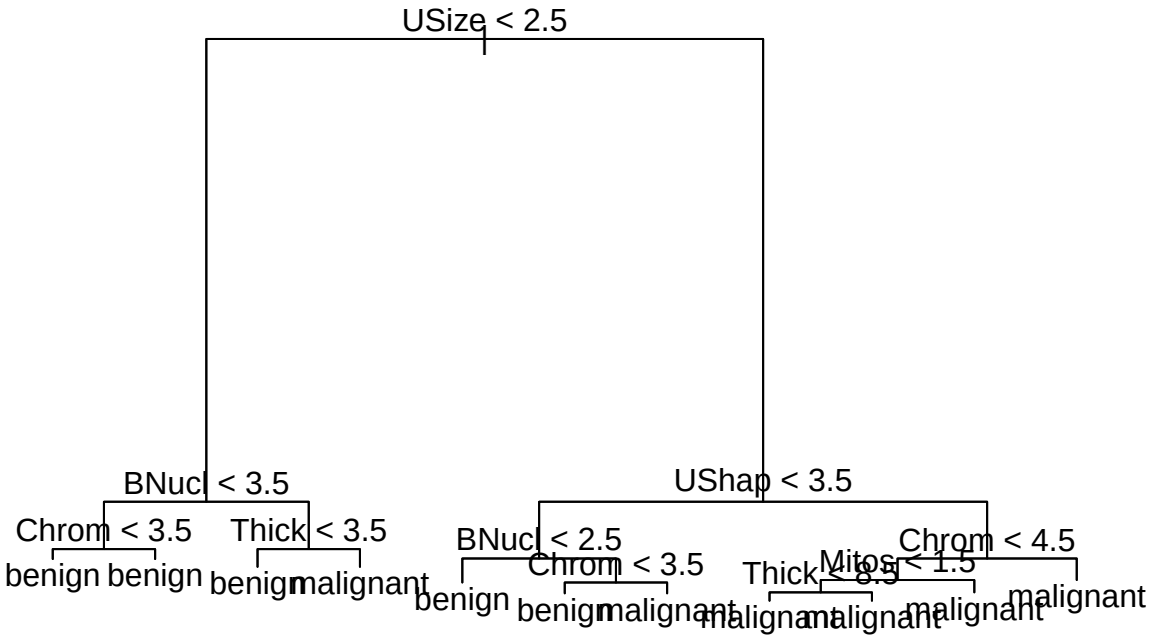
```
confmat_kable(
  cm_train_tree,
  caption = "Training confusion matrix: tree model"
)
```

Table 13: Training confusion matrix: tree model

Prediction	Reference	Freq
malignant	malignant	162
benign	malignant	5
malignant	benign	10
benign	benign	301

```
confmat_heatmap(
  cm_train_tree,
  title = "Training confusion matrix: tree model"
)
```





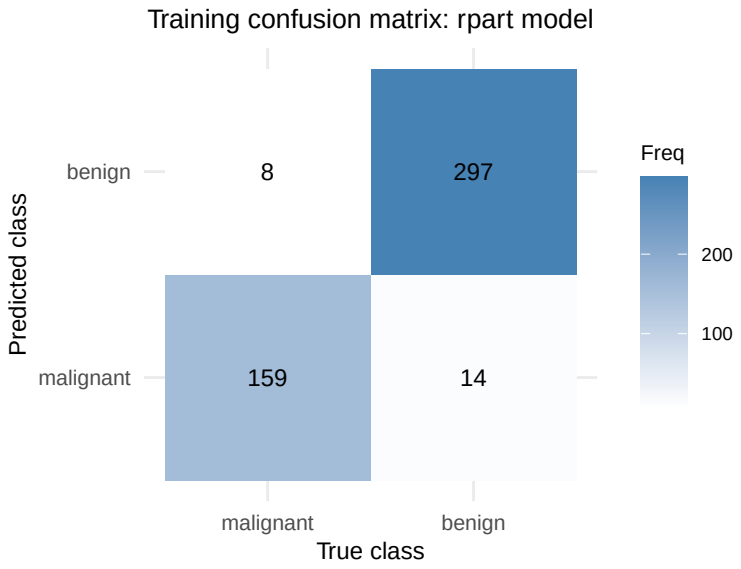
Tree model (rpart package) We next fit a classification tree using `rpart` with the Gini impurity criterion and compute the training confusion matrix.

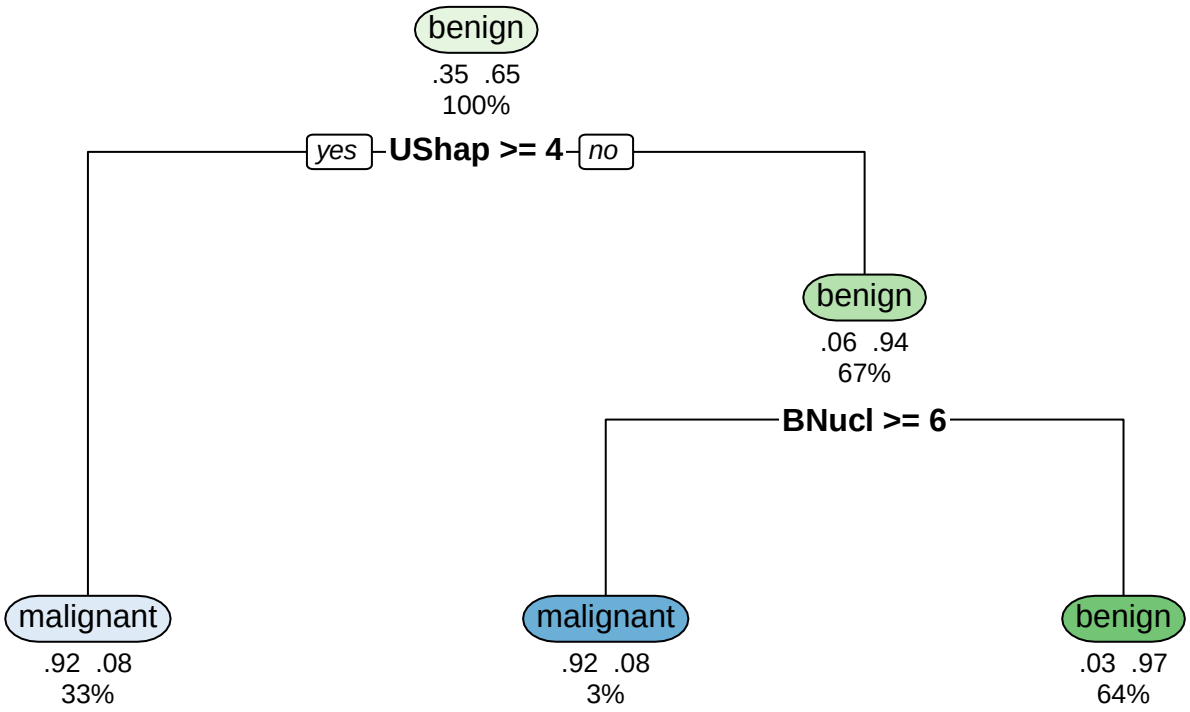
```
set.seed(1023)

rpart_model <- rpart::rpart(
  Class ~ .,
  data   = wbca_train,
  method = "class"
)

# Training predictions and confusion matrix
train_pred_rpart <- predict(
  rpart_model,
  newdata = wbca_train,
  type    = "class"
)

cm_train_rpart <- caret::confusionMatrix(
  data      = train_pred_rpart,
  reference = wbca_train_y,
  positive  = "malignant"
)

# Confusion matrix table + heatmap
confmat_kable(
  cm_train_rpart,
  caption = "Training confusion matrix: rpart model"
)
```

Table 14: Training confusion matrix: rpart model

Prediction	Reference	Freq
malignant	malignant	159
benign	malignant	8
malignant	benign	14
benign	benign	297

```
confmat_heatmap(
  cm_train_rpart,
  title = "Training confusion matrix: rpart model"
)
```





Random forest model We now fit a random forest classifier using the `randomForest` package and compute its training confusion matrix.

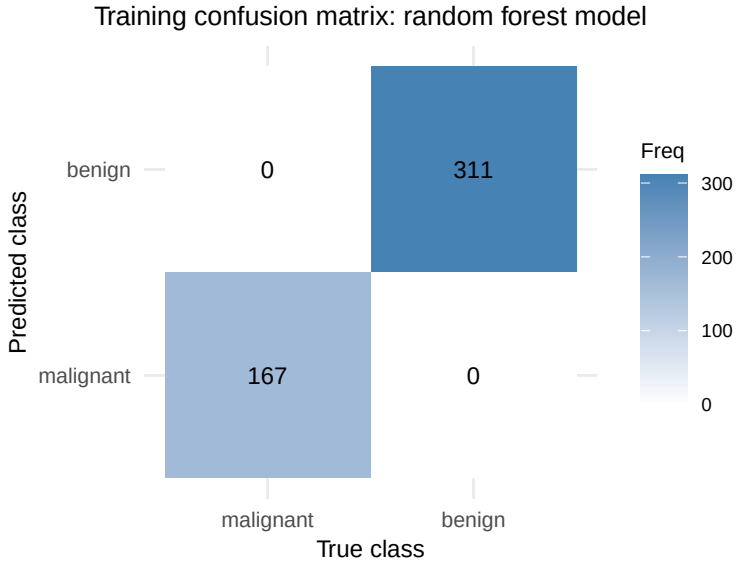
```
set.seed(1023)

rf_model <- randomForest::randomForest(
  Class ~ .,
  data = wbca_train
)

# Training predictions and confusion matrix
train_pred_rf <- predict(
  rf_model,
  newdata = wbca_train
)

cm_train_rf <- caret::confusionMatrix(
  data      = train_pred_rf,
  reference = wbca_train_y,
```

```
  positive  = "malignant"
)

# Confusion matrix table + heatmap
confmat_kable(
  cm_train_rf,
  caption = "Training confusion matrix: random forest model"
)
```

Table 15: Training confusion matrix: random forest model

Prediction	Reference	Freq
malignant	malignant	167
benign	malignant	0
malignant	benign	0
benign	benign	311

```
confmat_heatmap(
  cm_train_rf,
  title = "Training confusion matrix: random forest model"
)
```



Boosting with C5.0 We fit a boosting model using `C5.0` from the `C50` package, treating *Class* as a factor response, and compute the training confusion matrix.

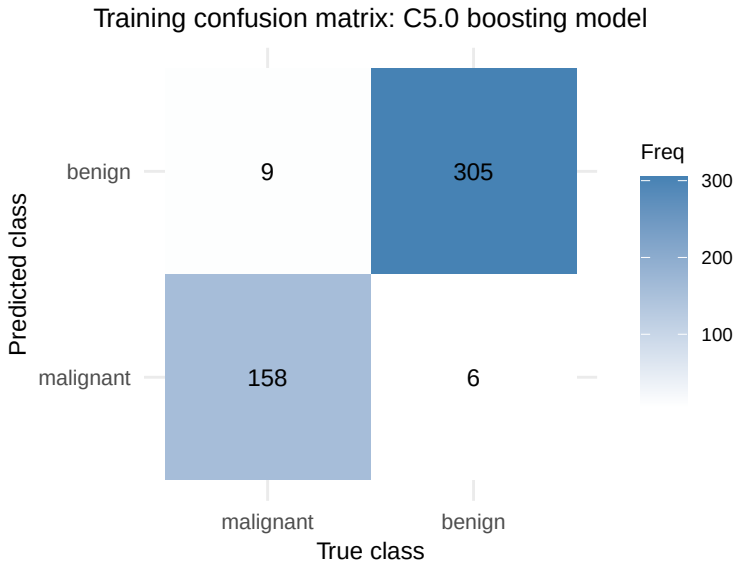
```
set.seed(1023)

c50_model <- C50::C5.0(
  x = wbca_train_x,
  y = wbca_train_y
)

# Training predictions and confusion matrix
train_pred_c50 <- predict(
  c50_model,
  newdata = wbca_train_x
)

cm_train_c50 <- caret::confusionMatrix(
  data      = train_pred_c50,
```

```
  reference = wbca_train_y,
  positive  = "malignant"
)

# Confusion matrix table + heatmap
confmat_kable(
  cm_train_c50,
  caption = "Training confusion matrix: C5.0 boosting model"
)
```

Table 16: Training confusion matrix: C5.0 boosting model

Prediction	Reference	Freq
malignant	malignant	158
benign	malignant	9
malignant	benign	6
benign	benign	305

```
confmat_heatmap(
  cm_train_c50,
  title = "Training confusion matrix: C5.0 boosting model"
)
```



AdaBoost model (adabag) We next fit an AdaBoost model using the `adabag` package with `Class` as the response and all nine predictors as covariates, then compute the training confusion matrix.

```
set.seed(1023)

# Lightweight AdaBoost so the whole Rmd knits quickly
ada_time <- system.time({
  ada_model <- adabag::boosting(
    Class   ~ .,
    data    = wbca_train,
    mfinal  = 10,  # keep this small (e.g. 5-10) for speed
    control = rpart::rpart.control(
      maxdepth = 2,    # very shallow trees
      minsplit = 20
    )
  )
})
```

```
ada_time  # so we can see how long it took when knitting
```

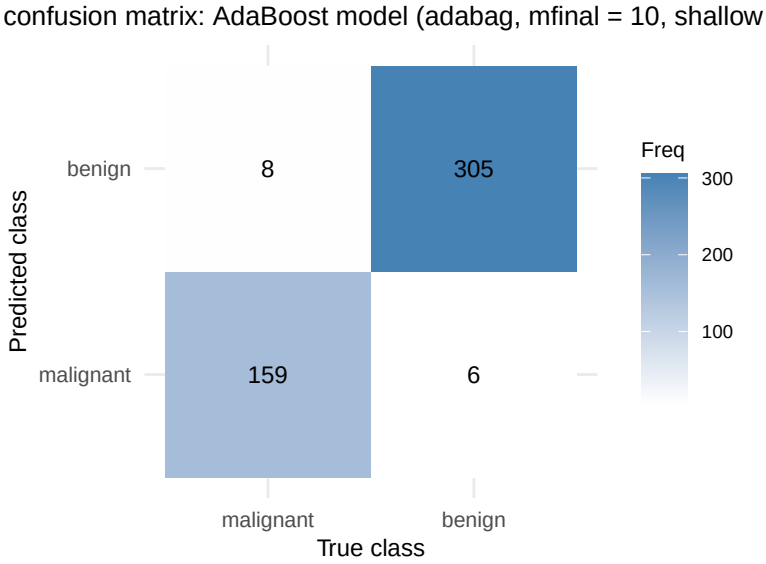
```
##    user  system elapsed
##   0.172   0.003   0.175
```

```
# Training predictions and confusion matrix
ada_train_pred <- predict(
  ada_model,
  newdata = wbca_train
)

train_pred_ada <- ada_train_pred$class

# Make sure both are factors with identical levels
wbca_train_y   <- factor(wbca_train_y)  # just to be sure
train_pred_ada <- factor(train_pred_ada, levels = levels(wbca_train_y))

cm_train_ada <- caret::confusionMatrix(
  data      = train_pred_ada,
  reference = wbca_train_y,
  positive  = "malignant"
)

# Confusion matrix table + heatmap
confmat_kable(
  cm_train_ada,
  caption = "Training confusion matrix: AdaBoost model (adabag, mfinal = 10, shallow trees)"
)
```

Table 17: Training confusion matrix: AdaBoost model (adabag, mfinal = 10, shallow trees)

Prediction	Reference	Freq
malignant	malignant	159
benign	malignant	8
malignant	benign	6
benign	benign	305

```
confmat_heatmap(
  cm_train_ada,
  title = "Training confusion matrix: AdaBoost model (adabag, mfinal = 10, shallow trees)"
)
```



XGBoost model Finally, we fit a gradient boosting model using `xgboost`. We first create a model matrix of predictors and a numeric response coded as 1 for benign and 0 for malignant, then fit a binary logistic model and classify using a probability cutoff of 0.5.

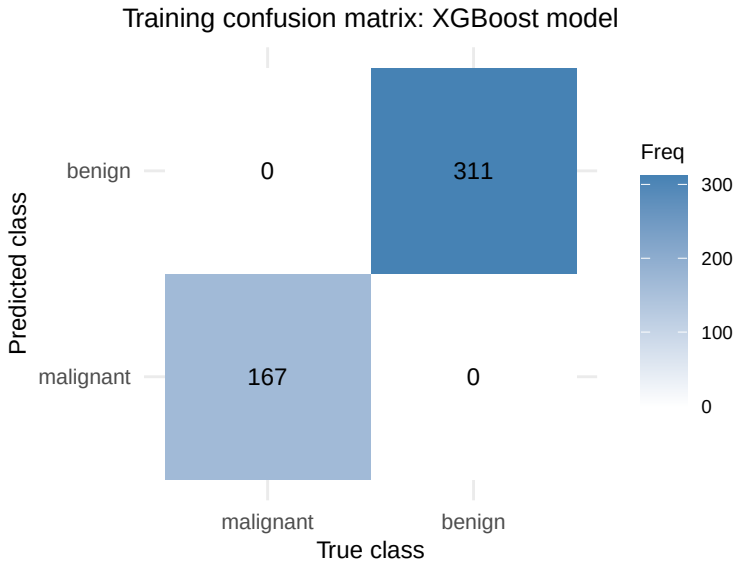
```
set.seed(1023)

# Model matrix for xgboost (no intercept column)
xgb_train_x <- model.matrix(
  Class ~ . - 1,
  data = wbca_train
)

xgb_train_y <- as.numeric(wbca_train_y == "benign")

xgb_model <- xgboost::xgboost(
  data      = xgb_train_x,
  label     = xgb_train_y,
  objective = "binary:logistic",
  nrounds   = 100,
  verbose   = 0
)

# Training predictions and confusion matrix
xgb_train_prob <- predict(
  xgb_model,
  newdata = xgb_train_x
)

train_pred_xgb <- ifelse(
  xgb_train_prob > 0.5,
  "benign",
  "malignant"
) %>%
  factor(levels = levels(wbca_train_y))

cm_train_xgb <- caret::confusionMatrix(
  data      = train_pred_xgb,
  reference = wbca_train_y,
  positive  = "malignant"
)
```

```r
# Confusion matrix table + heatmap
confmat_kable(
  cm_train_xgb,
  caption = "Training confusion matrix: XGBoost model"
)
```

Table 18: Training confusion matrix: XGBoost model

Prediction	Reference	Freq
malignant	malignant	167
benign	malignant	0
malignant	benign	0
benign	benign	311

```r
confmat_heatmap(
  cm_train_xgb,
  title = "Training confusion matrix: XGBoost model"
)
```



Conclusion: All six tree-based and ensemble methods achieve very strong performance on the training data, but with different degrees of aggressiveness and potential overfitting. The single-tree models (`tree` and `rpart`) have training accuracies in roughly the mid-90% range, similar to the reduced logistic regression model, with a reasonable balance between sensitivity and specificity for malignant tumors: for example, the `tree` fit slightly increases sensitivity for malignant cases relative to logistic regression while sacrificing a bit of specificity. The boosted models (C5.0 and AdaBoost) push training accuracy slightly higher and tend to reduce misclassification of malignant tumors further, indicating that they are capturing additional nonlinear interactions and threshold effects in the predictors. In contrast, the random forest and XGBoost models perfectly classify all training observations (100% accuracy, sensitivity, and specificity), which is a classic sign of potential overfitting: these flexible ensembles are powerful enough to effectively memorize the training set. Compared with the reduced logistic regression model, which already has excellent training performance but still makes a handful of errors, random forest and XGBoost clearly fit the training data more tightly. This makes it especially important to evaluate all models on the test set before deciding which one generalizes best.

2(b) Test-set confusion matrices and model comparison

We now evaluate each of the tree-based and ensemble models on the held-out test set. For each method, we compute predicted classes on `wbca_test`, form the confusion matrix, and compare the test performance to that of the reduced logistic regression model from Problem 1 (with the chosen cutoff). We then choose a champion model based on test-set performance and interpretability.

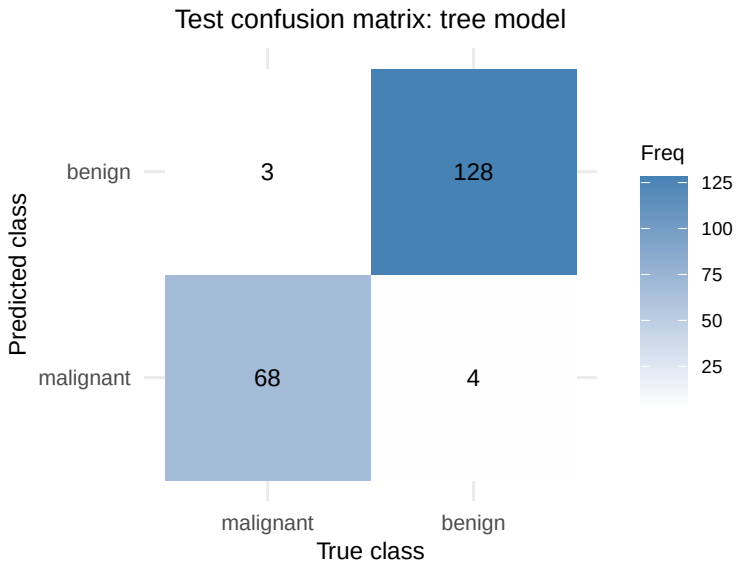
```r
# Test predictions and confusion matrix: tree model
test_pred_tree <- predict(
  tree_model,
  newdata = wbca_test,
  type    = "class"
)

cm_test_tree <- caret::confusionMatrix(
  data      = test_pred_tree,
  reference = wbca_test_y,
  positive  = "malignant"
)

# Confusion matrix table + heatmap
confmat_kable(
  cm_test_tree,
  caption = "Test confusion matrix: tree model"
)
```

Tree model (tree package) on the test set

Table 19: Test confusion matrix: tree model

Prediction	Reference	Freq
malignant	malignant	68
benign	malignant	3
malignant	benign	4
benign	benign	128

```r
confmat_heatmap(
  cm_test_tree,
  title = "Test confusion matrix: tree model"
)
```




```r
# Test predictions and confusion matrix: rpart model
test_pred_rpart <- predict(
  rpart_model,
  newdata = wbca_test,
```

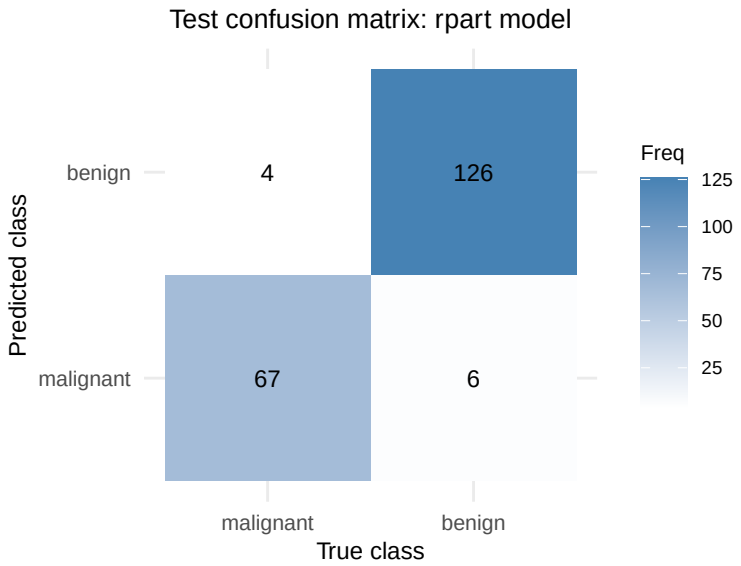
```
  type    = "class"
)

cm_test_rpart <- caret::confusionMatrix(
  data      = test_pred_rpart,
  reference = wbca_test_y,
  positive  = "malignant"
)

# Confusion matrix table + heatmap
confmat_kable(
  cm_test_rpart,
  caption = "Test confusion matrix: rpart model"
)
```

Tree model (rpart package) on the test set

Table 20: Test confusion matrix: rpart model

Prediction	Reference	Freq
malignant	malignant	67
benign	malignant	4
malignant	benign	6
benign	benign	126

```
confmat_heatmap(
  cm_test_rpart,
  title = "Test confusion matrix: rpart model"
)
```




```
# Test predictions and confusion matrix: random forest model
test_pred_rf <- predict(
  rf_model,
  newdata = wbca_test
)

cm_test_rf <- caret::confusionMatrix(
  data      = test_pred_rf,
  reference = wbca_test_y,
```

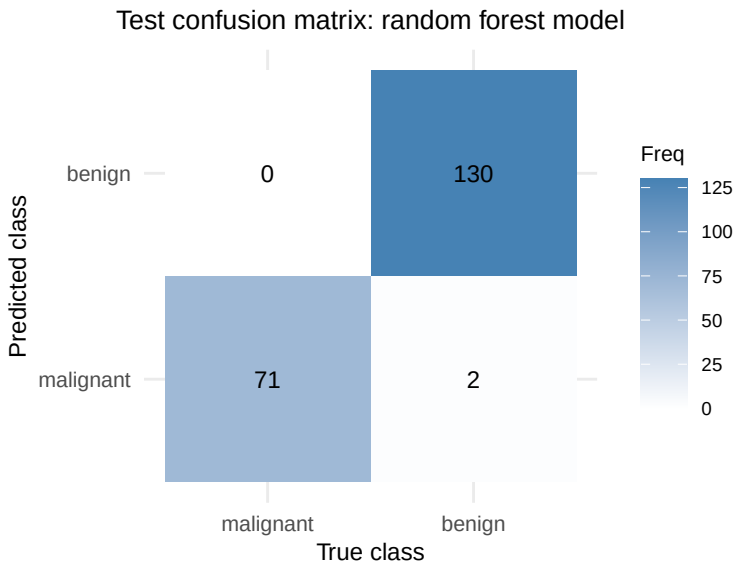
```
  positive  = "malignant"
)

# Confusion matrix table + heatmap
confmat_kable(
  cm_test_rf,
  caption = "Test confusion matrix: random forest model"
)
```

Random forest model on the test set

Table 21: Test confusion matrix: random forest model

Prediction	Reference	Freq
malignant	malignant	71
benign	malignant	0
malignant	benign	2
benign	benign	130

```
confmat_heatmap(
  cm_test_rf,
  title = "Test confusion matrix: random forest model"
)
```




```
# Test predictions and confusion matrix: C5.0 boosting model
test_pred_c50 <- predict(
  c50_model,
  newdata = wbca_test_x
)

cm_test_c50 <- caret::confusionMatrix(
  data      = test_pred_c50,
  reference = wbca_test_y,
  positive  = "malignant"
)

# Confusion matrix table + heatmap
confmat_kable(
  cm_test_c50,
```

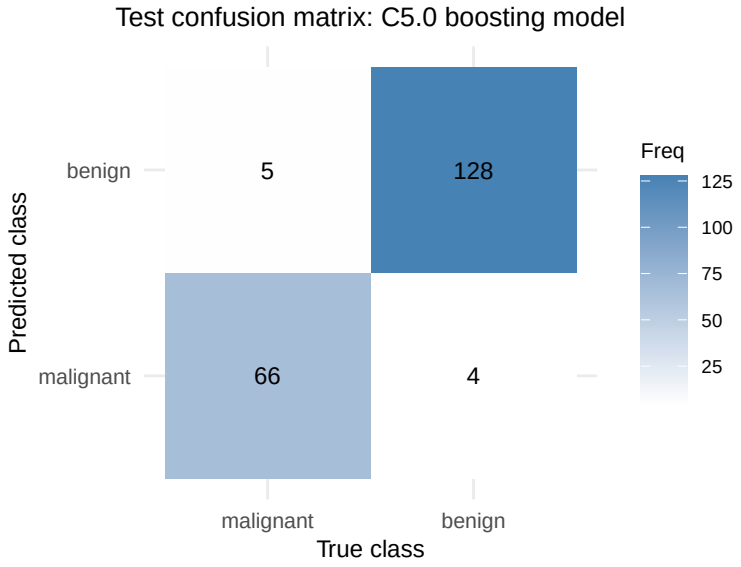
```
  caption = "Test confusion matrix: C5.0 boosting model"
)
```

C5.0 boosting model on the test set

Table 22: Test confusion matrix: C5.0 boosting model

Prediction	Reference	Freq
malignant	malignant	66
benign	malignant	5
malignant	benign	4
benign	benign	128

```
confmat_heatmap(
  cm_test_c50,
  title = "Test confusion matrix: C5.0 boosting model"
)
```



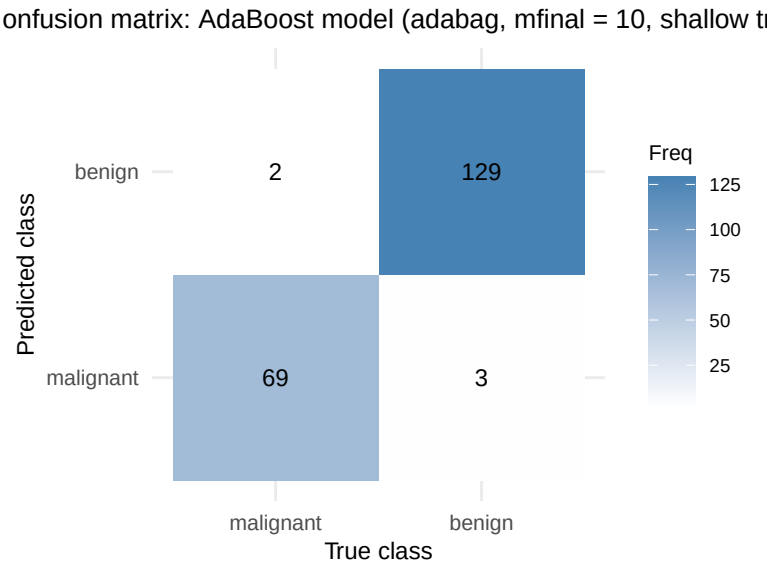

```
# Test predictions and confusion matrix: AdaBoost model (adabag)
ada_test_pred <- predict(
  ada_model,
  newdata = wbca_test
)

test_pred_ada <- ada_test_pred$class
test_pred_ada <- factor(test_pred_ada, levels = levels(wbca_test_y))

cm_test_ada <- caret::confusionMatrix(
  data      = test_pred_ada,
  reference = wbca_test_y,
  positive  = "malignant"
)

# Confusion matrix table + heatmap
confmat_kable(
  cm_test_ada,
  caption = "Test confusion matrix: AdaBoost model (adabag, mfinal = 10, shallow trees)"
)
```

AdaBoost model (adabag) on the test set

Table 23: Test confusion matrix: AdaBoost model (adabag, mfinal = 10, shallow trees)

Prediction	Reference	Freq
malignant	malignant	69
benign	malignant	2
malignant	benign	3
benign	benign	129

```
confmat_heatmap(
  cm_test_ada,
  title = "Test confusion matrix: AdaBoost model (adabag, mfinal = 10, shallow trees)"
)
```




```
# Test predictions and confusion matrix: XGBoost model
xgb_test_x <- model.matrix(
  Class ~ . - 1,
  data = wbca_test
)

xgb_test_prob <- predict(
  xgb_model,
  newdata = xgb_test_x
)

test_pred_xgb <- ifelse(
  xgb_test_prob > 0.5,
  "benign",
  "malignant"
) %>%
  factor(levels = levels(wbca_test_y))

cm_test_xgb <- caret::confusionMatrix(
  data      = test_pred_xgb,
  reference = wbca_test_y,
  positive  = "malignant"
)

# Confusion matrix table + heatmap
confmat_kable(
```

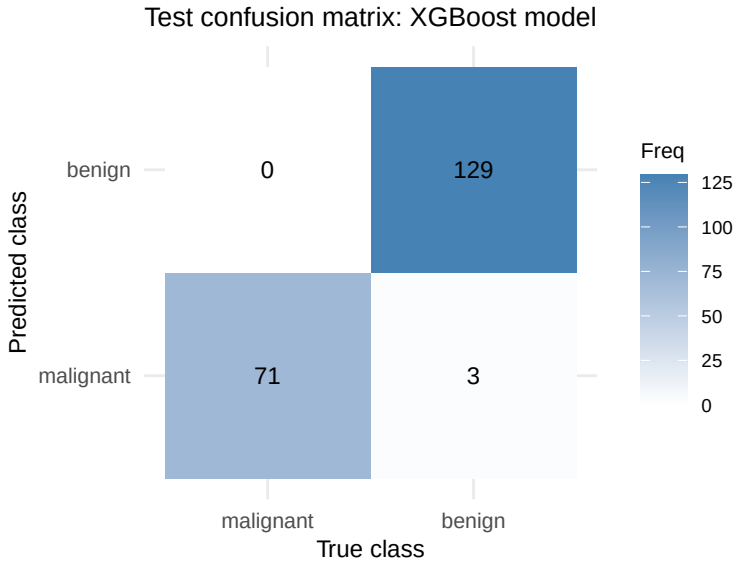
```
  cm_test_xgb,
  caption = "Test confusion matrix: XGBoost model"
)
```

XGBoost model on the test set

Table 24: Test confusion matrix: XGBoost model

Prediction	Reference	Freq
malignant	malignant	71
benign	malignant	0
malignant	benign	3
benign	benign	129

```
confmat_heatmap(
  cm_test_xgb,
  title = "Test confusion matrix: XGBoost model"
)
```



Summary of test-set performance across methods To help compare all methods, including logistic regression, we assemble a summary table of test accuracy, sensitivity, and specificity for malignant cases. For logistic regression, we include both cutoffs 0.5 and 0.9; for the tree-based methods, we use the default cutoff implied by their predicted classes.

```
test_performance_summary <- tibble(
  model = c(
    "Logistic (cutoff 0.5)",
    "Logistic (cutoff 0.9)",
    "Tree",
    "rpart",
    "Random forest",
    "C5.0",
    "AdaBoost",
    "XGBoost"
  ),
  accuracy = c(
    cm_test_cut_05$overall["Accuracy"],
    cm_test_cut_09$overall["Accuracy"],
    cm_test_tree$overall["Accuracy"],
    cm_test_rpart$overall["Accuracy"],
    cm_test_rf$overall["Accuracy"],
    cm_test_c50$overall["Accuracy"],
```

```
    cm_test_ada$overall["Accuracy"],
    cm_test_xgb$overall["Accuracy"]
  ),
  sensitivity_malignant = c(
    cm_test_cut_05$byClass["Sensitivity"],
    cm_test_cut_09$byClass["Sensitivity"],
    cm_test_tree$byClass["Sensitivity"],
    cm_test_rpart$byClass["Sensitivity"],
    cm_test_rf$byClass["Sensitivity"],
    cm_test_c50$byClass["Sensitivity"],
    cm_test_ada$byClass["Sensitivity"],
    cm_test_xgb$byClass["Sensitivity"]
  ),
  specificity_malignant = c(
    cm_test_cut_05$byClass["Specificity"],
    cm_test_cut_09$byClass["Specificity"],
    cm_test_tree$byClass["Specificity"],
    cm_test_rpart$byClass["Specificity"],
    cm_test_rf$byClass["Specificity"],
    cm_test_c50$byClass["Specificity"],
    cm_test_ada$byClass["Specificity"],
    cm_test_xgb$byClass["Specificity"]
  )
)

knitr::kable(
  test_performance_summary,
  digits  = 4,
  caption = "Test-set accuracy, sensitivity, and specificity for malignant cases across all models"
)
```

Table 25: Test-set accuracy, sensitivity, and specificity for malignant cases across all models

model	accuracy	sensitivity_malignant	specificity_malignant
Logistic (cutoff 0.5)	0.9852	1.0000	0.9773
Logistic (cutoff 0.9)	0.9803	1.0000	0.9697
Tree	0.9655	0.9577	0.9697
rpart	0.9507	0.9437	0.9545
Random forest	0.9901	1.0000	0.9848
C5.0	0.9557	0.9296	0.9697
AdaBoost	0.9754	0.9718	0.9773
XGBoost	0.9852	1.0000	0.9773

Conclusion: The test-set summary shows that several models perform exceptionally well on the wbca data. The reduced logistic regression with cutoff 0.5 achieves accuracy around 0.985, with perfect sensitivity for malignant tumors (1.000) and specificity about 0.977. XGBoost attains essentially identical test performance (accuracy 0.985, sensitivity 1.000, specificity 0.977), while the random forest edges out all other methods with the highest test accuracy (about 0.990) and similarly perfect sensitivity and slightly higher specificity (about 0.985). The single-tree models and the other boosted ensembles (C5.0 and AdaBoost) trail these leaders by a small margin, with accuracies in the mid- to high-0.95 range and sensitivities for malignant cases between roughly 0.93 and 0.97.

Comparing training and test performance, random forest and XGBoost show the most dramatic gap: they achieve literally perfect classification on the training data, yet their test accuracies, while still excellent, drop slightly below perfection. This pattern indicates that these models are flexible enough to memorize the training set, though in this example the overfitting is mild because their test performance remains among the best. Simpler models such as logistic regression and the single-tree fits exhibit more modest training accuracy but maintain very strong performance on the test set, suggesting good generalization.

If the goal is to choose a single "champion" model purely on predictive performance, the random forest is the natural choice: it offers the highest test accuracy and perfect sensitivity for malignant tumors while retaining very high specificity,

minimizing both missed cancers and unnecessary alarms. However, the reduced logistic regression model with cutoff 0.5 is extremely competitive—only slightly worse in test accuracy and specificity—and has the major advantage of interpretability: its coefficients directly quantify how each predictor affects the log-odds of benign vs malignant status. In practice, one might use the logistic model as the primary, interpretable screening tool and regard the random forest as a slightly more accurate but less transparent alternative.

```
## Total time to run: 6.1 secs
```