Titanic Survival – Challenger Models (Logistic Regression & Decision Tree)

Orkhon.Kh

2025-11-27

V. Challenger Models: Logistic Regression and Decision Tree

This section presents two **challenger classification models** for predicting whether a passenger survived the sinking of the RMS Titanic:

- **Logistic regression** (parametric benchmark model)
- **Classification tree** (non-parametric regression tree for a binary outcome)

Both models are built on the same cleaned training data set and evaluated on a held-out test set using in-sample and out-of-sample performance metrics.

1. Data Preparation

We start from the Titanic survival data file provided by the instructor, assumed to be in the working directory as `Titanic_Survival_Data.csv`. We keep the variables that are plausibly useful for prediction and drop high-cardinality identifiers such as name, ticket, cabin, boat, body and home destination.

```
titanic_raw <- read.csv("Titanic_Survival_Data.csv")

# Keep core modeling variables
titanic <- titanic_raw |>
  select(
    survived,
    pclass,
    sex,
    age,
    sibsp,
    parch,
    fare,
    embarked
  )

str(titanic)
```

```
## 'data.frame':    1310 obs. of  8 variables:
##  $ survived: int  1 1 0 0 0 1 1 0 1 0 ...
##  $ pclass  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ sex     : chr  "female" "male" "female" "male" ...
##  $ age     : num  29 0.917 2 30 25 ...
##  $ sibsp   : int  0 1 1 1 1 0 1 0 2 0 ...
##  $ parch   : int  0 2 2 2 2 0 0 0 0 0 ...
##  $ fare    : num  211 152 152 152 152 ...
```

```
##  $ embarked: chr  "S" "S" "S" "S" ...
```

```
summary(titanic)
```

```
##     survived         pclass          sex                 age
##  Min.   :0.000   Min.   :1.000   Length:1310        Min.   : 0.1667
##  1st Qu.:0.000   1st Qu.:2.000   Class :character   1st Qu.:21.0000
##  Median :0.000   Median :3.000   Mode  :character   Median :28.0000
##  Mean   :0.382   Mean   :2.295                      Mean   :29.8811
##  3rd Qu.:1.000   3rd Qu.:3.000                      3rd Qu.:39.0000
##  Max.   :1.000   Max.   :3.000                      Max.   :80.0000
##  NA's   :1       NA's   :1                          NA's   :264
##      sibsp            parch            fare           embarked
##  Min.   :0.0000   Min.   :0.000   Min.   :  0.000   Length:1310
##  1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:  7.896   Class :character
##  Median :0.0000   Median :0.000   Median : 14.454   Mode  :character
##  Mean   :0.4989   Mean   :0.385   Mean   : 33.295
##  3rd Qu.:1.0000   3rd Qu.:0.000   3rd Qu.: 31.275
##  Max.   :8.0000   Max.   :9.000   Max.   :512.329
##  NA's   :1        NA's   :1       NA's   :2
```

There are missing values in `age`, `fare`, and `embarked`. For this project, and to keep the challenger models simple and transparent, we restrict attention to **complete cases** on these variables.

```
titanic_complete <- titanic |>
  drop_na()

# Convert appropriate variables to factors
titanic_complete <- titanic_complete |>
  mutate(
    survived = factor(survived, levels = c(0, 1), labels = c("No", "Yes")),
    pclass   = factor(pclass, levels = c(1, 2, 3),
                      labels = c("1st", "2nd", "3rd")),
    sex      = factor(sex),
    embarked = factor(embarked)
  )

dim(titanic_complete)
```

```
## [1] 1045    8
```

```
summary(titanic_complete)
```

```
##  survived  pclass        sex           age              sibsp
##  No :618   1st:284   female:388   Min.   : 0.1667   Min.   :0.0000
##  Yes:427   2nd:261   male  :657   1st Qu.:21.0000   1st Qu.:0.0000
##            3rd:500                Median :28.0000   Median :0.0000
##                                   Mean   :29.8518   Mean   :0.5033
##                                   3rd Qu.:39.0000   3rd Qu.:1.0000
##                                   Max.   :80.0000   Max.   :8.0000
##      parch            fare        embarked
##  Min.   :0.0000   Min.   :  0.00    :  2
##  1st Qu.:0.0000   1st Qu.:  8.05   C:212
##  Median :0.0000   Median : 15.75   Q: 50
##  Mean   :0.4211   Mean   : 36.69   S:781
##  3rd Qu.:1.0000   3rd Qu.: 35.50
##  Max.   :6.0000   Max.   :512.33
```

2. Train–Test Split

Following the project instructions, we use a **70% / 30% split** with `set.seed(1023)` to ensure reproducibility. The **train set** is used for model fitting and selection; the **test set** is used only for final performance assessment.

```
set.seed(1023)

n_total <- nrow(titanic_complete)
train_index <- sample(1:n_total, size = 0.7 * n_total)

train <- titanic_complete[train_index, ]
test  <- titanic_complete[-train_index, ]

dim(train)
```

```
## [1] 731   8
```

```
dim(test)
```

```
## [1] 314   8
```

3. Helper Function: Classification Metrics

To compare challenger models on a common basis, we define a simple helper function that takes:

- the **true outcome** (`actual`, either numeric 0/1 or factor with levels "No"/"Yes"), and
- the **predicted survival probability** (`prob` = Pr(Survived = Yes))

and returns:

- confusion matrix at a probability cut-off (default 0.5)
- accuracy
- sensitivity (true positive rate for survivors)
- specificity (true negative rate for non-survivors)
- AUC (area under the ROC curve)

```
classification_metrics <- function(actual, prob, cutoff = 0.5) {
  # Convert factor responses to numeric 0/1 with "Yes" as 1
  if (is.factor(actual)) {
    # Assume the second level corresponds to "Yes"
    positive_level <- levels(actual)[2]
    actual_num <- ifelse(actual == positive_level, 1, 0)
  } else {
    actual_num <- actual
  }

  pred_class_num <- ifelse(prob >= cutoff, 1, 0)

  cm <- table(
    Predicted = pred_class_num,
    Actual    = actual_num
  )

  accuracy <- sum(diag(cm)) / sum(cm)

  # Guard against division by zero in edge cases
  sens <- ifelse(sum(cm[, "1"]) == 0, NA, cm["1", "1"] / sum(cm[, "1"]))
```

```
  spec <- ifelse(sum(cm[, "0"]) == 0, NA, cm["0", "0"] / sum(cm[, "0"]))

  roc_obj <- roc(response = actual_num, predictor = prob, quiet = TRUE)
  auc_val <- as.numeric(auc(roc_obj))

  metrics_tbl <- tibble(
    Accuracy    = accuracy,
    Sensitivity = sens,
    Specificity = spec,
    AUC         = auc_val
  )

  list(
    confusion_matrix = cm,
    metrics          = metrics_tbl
  )
}
```

4. Logistic Regression Challenger Model

4.1 Model Specification and Variable Selection

We first fit a **full logistic regression model** with all available predictors, and then apply stepwise AIC selection to obtain a more parsimonious challenger model.

```
# Full model with main effects only
logit_full <- glm(
  survived ~ pclass + sex + age + sibsp + parch + fare + embarked,
  family = binomial,
  data   = train
)

summary(logit_full)
```

```
##
## Call:
## glm(formula = survived ~ pclass + sex + age + sibsp + parch +
##     fare + embarked, family = binomial, data = train)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.684e+01  6.093e+02   0.028  0.97795
## pclass2nd   -1.132e+00  3.320e-01  -3.410  0.00065 ***
## pclass3rd   -2.241e+00  3.436e-01  -6.521 6.96e-11 ***
## sexmale     -2.919e+00  2.280e-01 -12.801  < 2e-16 ***
## age         -4.275e-02  8.351e-03  -5.119 3.07e-07 ***
## sibsp       -4.126e-01  1.417e-01  -2.911  0.00360 **
## parch        1.534e-01  1.328e-01   1.155  0.24808
## fare        -7.506e-04  2.401e-03  -0.313  0.75456
## embarkedC   -1.203e+01  6.093e+02  -0.020  0.98425
## embarkedQ   -1.398e+01  6.093e+02  -0.023  0.98170
## embarkedS   -1.273e+01  6.093e+02  -0.021  0.98333
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 995.21  on 730  degrees of freedom
## Residual deviance: 621.28  on 720  degrees of freedom
## AIC: 643.28
##
## Number of Fisher Scoring iterations: 13
```

```
# Stepwise selection (both directions) based on AIC
logit_step <- step(logit_full, direction = "both", trace = FALSE)

summary(logit_step)
```

```
##
## Call:
## glm(formula = survived ~ pclass + sex + age + sibsp + embarked,
##     family = binomial, data = train)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  16.792107 609.151548   0.028  0.97801
## pclass2nd    -1.101881   0.303310  -3.633  0.00028 ***
## pclass3rd    -2.195269   0.299178  -7.338 2.17e-13 ***
## sexmale      -2.952392   0.225943 -13.067  < 2e-16 ***
## age          -0.042950   0.008326  -5.159 2.49e-07 ***
## sibsp        -0.366446   0.132861  -2.758  0.00581 **
## embarkedC   -11.971045 609.151459  -0.020  0.98432
## embarkedQ   -13.933448 609.151608  -0.023  0.98175
## embarkedS   -12.664490 609.151443  -0.021  0.98341
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 995.21  on 730  degrees of freedom
## Residual deviance: 622.61  on 722  degrees of freedom
## AIC: 640.61
##
## Number of Fisher Scoring iterations: 13
```

The stepwise procedure the strongest predictors such as passenger class, sex, and age, while potentially dropping weaker variables that do not materially improve model fit.

4.2 Multicollinearity Check

As a basic diagnostic for multicollinearity, we compute **variance inflation factors (VIFs)** for the selected logistic model.

```
vif(logit_step)
```

```
##              GVIF Df GVIF^(1/(2*Df))
## pclass   1.640518  2        1.131736
## sex      1.179172  1        1.085897
## age      1.442655  1        1.201106
## sibsp    1.112923  1        1.054951
## embarked 1.267963  3        1.040362
```

The multicollinearity check confirms that the selected logistic regression model does not suffer from problematic predictor correlation. This supports the validity of the model's coefficient estimates and ensures that the relationships between predictors and survival outcomes can be interpreted with confidence.

4.3 In-Sample Performance (Train Set)

```
# Predicted survival probabilities on the train set
logit_train_prob <- predict(logit_step, newdata = train, type = "response")

# Evaluate at cut-off = 0.5
logit_train_eval <- classification_metrics(train$survived, logit_train_prob, cutoff = 0.5)

# Confusion matrix
logit_train_eval$confusion_matrix
```

```
##           Actual
## Predicted   0   1
##         0 371  87
##         1  52 221
```

```
# Summary metrics
kable(
  logit_train_eval$metrics,
  digits = 3,
  caption = "Logistic Regression - Training Performance (cut-off = 0.5)"
)
```

Table 1: Logistic Regression – Training Performance (cut-off = 0.5)

Accuracy	Sensitivity	Specificity	AUC
0.81	0.718	0.877	0.874

4.4 Out-of-Sample Performance (Test Set)

```
# Predicted probabilities on the test set
logit_test_prob <- predict(logit_step, newdata = test, type = "response")

logit_test_eval <- classification_metrics(test$survived, logit_test_prob, cutoff = 0.5)

# Confusion matrix
logit_test_eval$confusion_matrix
```

```
##           Actual
## Predicted   0   1
##         0 156  39
##         1  39  80
```

```
# Summary metrics
kable(
  logit_test_eval$metrics,
  digits = 3,
  caption = "Logistic Regression - Test Performance (cut-off = 0.5)"
)
```

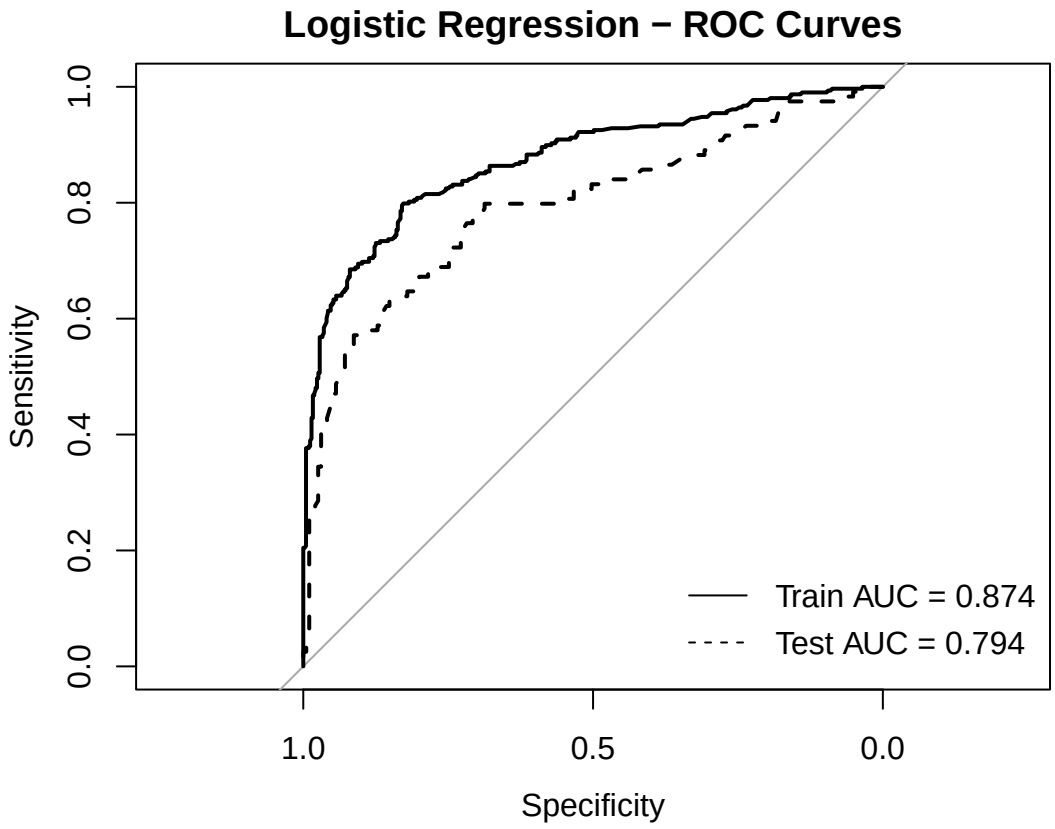
Table 2: Logistic Regression – Test Performance (cut-off = 0.5)

Accuracy	Sensitivity	Specificity	AUC
0.752	0.672	0.8	0.794

The logistic regression model demonstrates strong and stable generalization when evaluated on the test dataset. Test accuracy (0.752) and AUC (0.794) remain close to the training results, indicating minimal overfitting. Sensitivity (0.672) shows the model is reasonably effective at identifying survivors, while specificity (0.800) confirms strong performance in correctly identifying non-survivors. Overall, the logistic regression model maintains consistent predictive ability across data splits and performs reliably as a challenger model.

4.5 ROC Curves (Optional Visualization)

```
roc_train_logit <- roc(response = train$survived, predictor = logit_train_prob, quiet = TRUE)
roc_test_logit  <- roc(response = test$survived,  predictor = logit_test_prob,  quiet = TRUE)

plot(roc_train_logit, main = "Logistic Regression - ROC Curves")
lines(roc_test_logit, lty = 2)
legend(
  "bottomright",
  legend = c(
    paste0("Train AUC = ", round(auc(roc_train_logit), 3)),
    paste0("Test AUC = ", round(auc(roc_test_logit), 3))
  ),
  lty = c(1, 2),
  bty = "n"
)
```



5. Decision Tree (Classification Tree) Challenger Model

As a non-parametric alternative, we build a **classification tree** to model survival as a function of the same set of predictors. Trees can capture non-linearities and interaction effects automatically, at the cost of potentially higher variance.

5.1 Fit a Full Tree

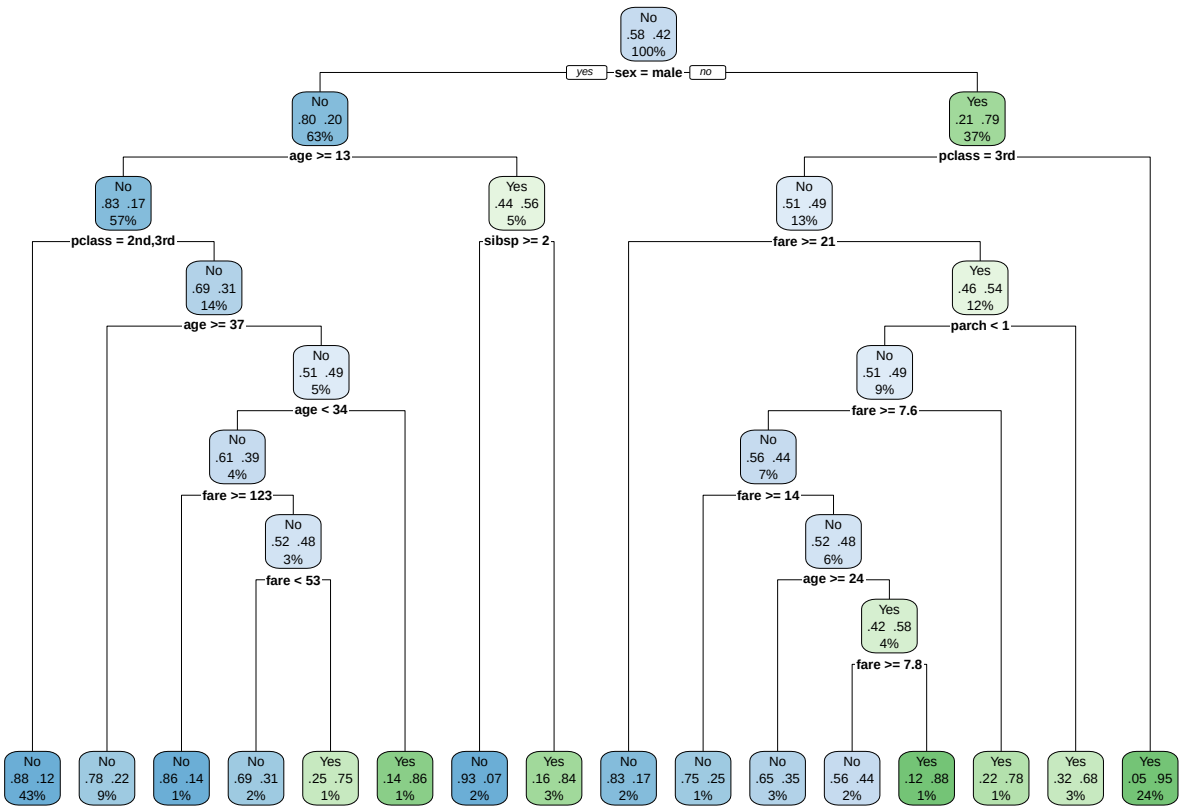
```
set.seed(1023)

tree_full <- rpart(
  survived ~ pclass + sex + age + sibsp + parch + fare + embarked,
  data   = train,
  method = "class",
  control = rpart.control(cp = 0.001)   # small cp to allow a reasonably large tree
)

# Complexity parameter table (cross-validated error)
printcp(tree_full)
```

```
##
## Classification tree:
## rpart(formula = survived ~ pclass + sex + age + sibsp + parch +
##     fare + embarked, data = train, method = "class", control = rpart.control(cp = 0.001))
##
## Variables actually used in tree construction:
## [1] age    fare   parch  pclass sex    sibsp
```

```
##
## Root node error: 308/731 = 0.42134
##
## n= 731
##
##          CP nsplit rel error  xerror     xstd
## 1 0.5194805      0   1.00000 1.00000 0.043345
## 2 0.0275974      1   0.48052 0.48052 0.035274
## 3 0.0129870      3   0.42532 0.43182 0.033866
## 4 0.0097403      5   0.39935 0.44805 0.034352
## 5 0.0064935      7   0.37987 0.44156 0.034160
## 6 0.0054113     10   0.36039 0.45130 0.034448
## 7 0.0010000     15   0.33117 0.43506 0.033965
```

```r
rpart.plot(
  tree_full,
  type = 2,
  extra = 104,              # display class, prob, and percentage of observations
  fallen.leaves = TRUE,
  main = "Full Classification Tree - Titanic Survival"
)
```

Full Classification Tree − Titanic Survival



5.2 Prune the Tree Using Cross-Validation

We use the cross-validated **relative error** (`xerror`) from the CP table to select a simpler subtree with better generalization.

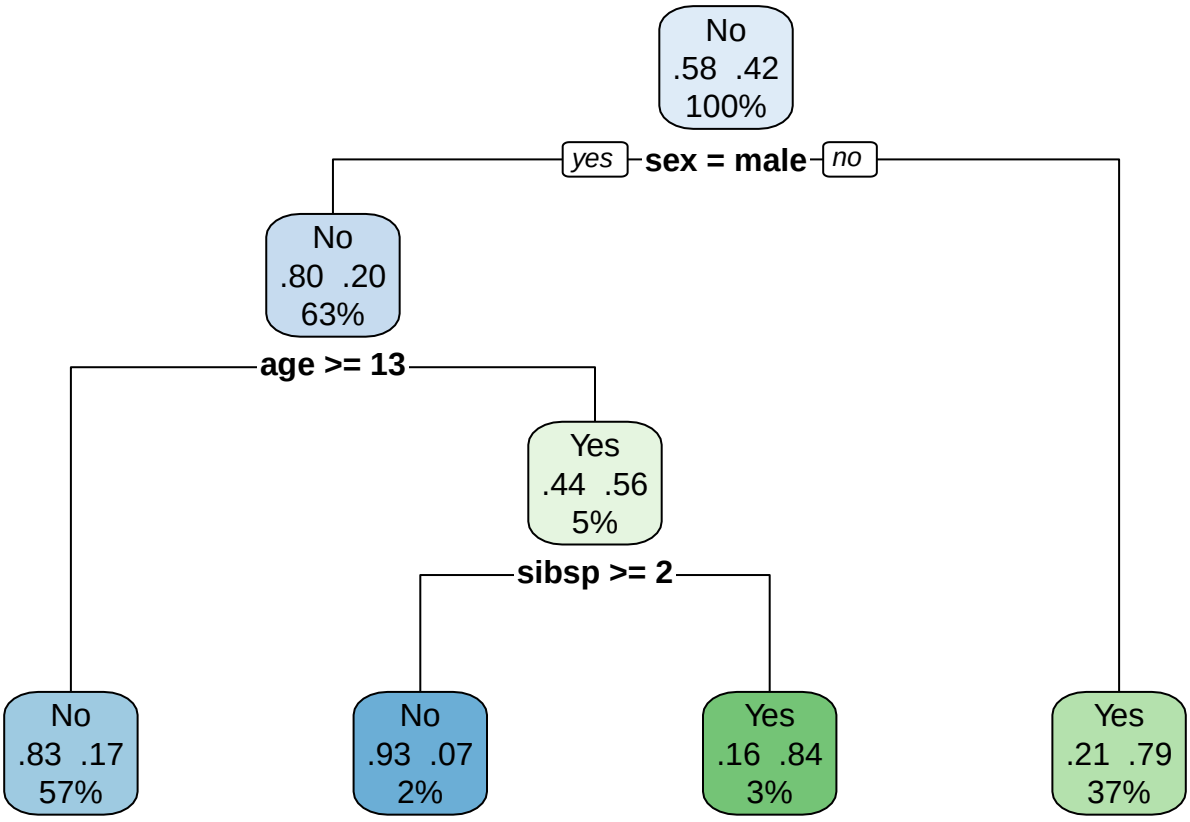
```
# Choose the cp value that minimizes cross-validated error
best_cp <- tree_full$cptable[which.min(tree_full$cptable[, "xerror"]), "CP"]
best_cp
```

```
## [1] 0.01298701
```

```
tree_pruned <- prune(tree_full, cp = best_cp)

tree_pruned
```

```
## n= 731
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 731 308 No (0.57865937 0.42134063)
##    2) sex=male 457  91 No (0.80087527 0.19912473)
##      4) age>=12.5 418  69 No (0.83492823 0.16507177) *
##      5) age< 12.5 39  17 Yes (0.43589744 0.56410256)
##       10) sibsp>=2 14   1 No (0.92857143 0.07142857) *
##       11) sibsp< 2 25   4 Yes (0.16000000 0.84000000) *
##    3) sex=female 274  57 Yes (0.20802920 0.79197080) *
```

```
rpart.plot(
  tree_pruned,
  type = 2,
  extra = 104,
  fallen.leaves = TRUE,
  main = "Pruned Classification Tree - Titanic Survival"
)
```



5.3 In-Sample Performance (Train Set)

```
# Predicted class probabilities for the "Yes" class
tree_train_prob <- predict(tree_pruned, newdata = train, type = "prob")[, "Yes"]

tree_train_eval <- classification_metrics(train$survived, tree_train_prob, cutoff = 0.5)

# Confusion matrix
tree_train_eval$confusion_matrix
```

```
##           Actual
## Predicted   0   1
##         0 362  70
##         1  61 238
```

```
# Summary metrics
kable(
  tree_train_eval$metrics,
  digits = 3,
  caption = "Classification Tree - Training Performance (cut-off = 0.5)"
)
```

Table 3: Classification Tree – Training Performance (cut-off = 0.5)

Accuracy	Sensitivity	Specificity	AUC
0.821	0.773	0.856	0.818

5.4 Out-of-Sample Performance (Test Set)

```
tree_test_prob <- predict(tree_pruned, newdata = test, type = "prob")[, "Yes"]

tree_test_eval <- classification_metrics(test$survived, tree_test_prob, cutoff = 0.5)

# Confusion matrix
tree_test_eval$confusion_matrix
```

```
##           Actual
## Predicted   0   1
##         0 155  40
##         1  40  79
```

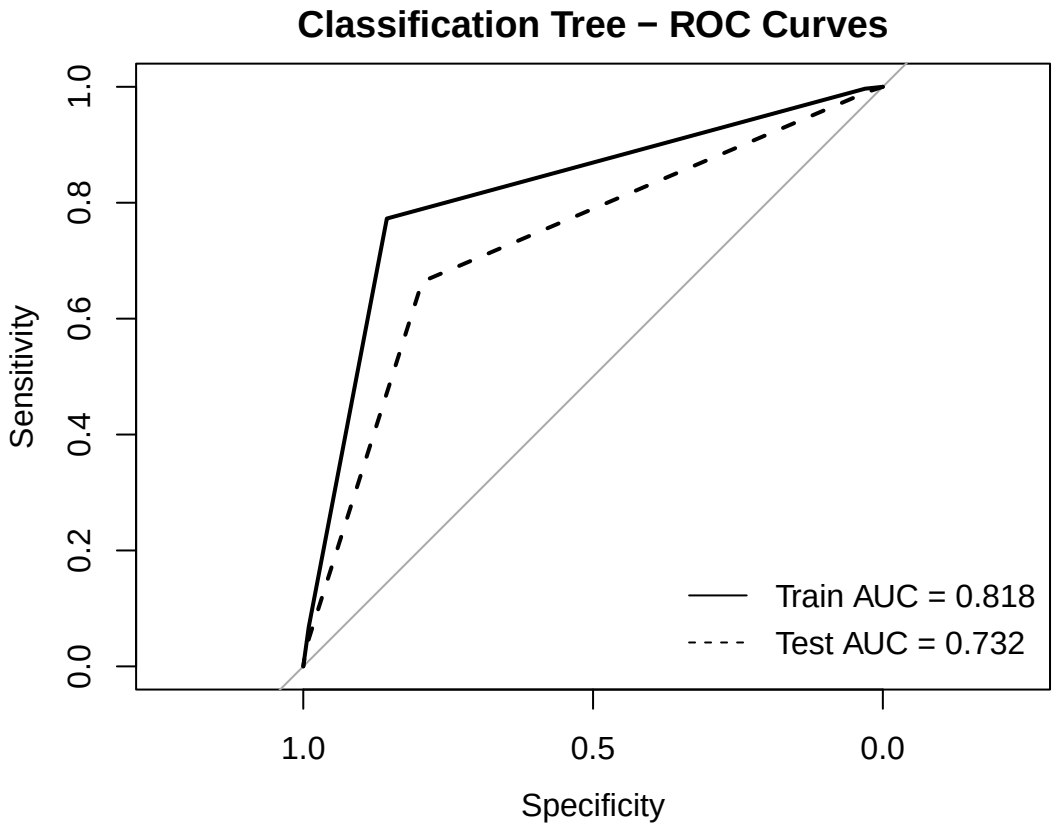
```
# Summary metrics
kable(
  tree_test_eval$metrics,
  digits = 3,
  caption = "Classification Tree - Test Performance (cut-off = 0.5)"
)
```

Table 4: Classification Tree – Test Performance (cut-off = 0.5)

Accuracy	Sensitivity	Specificity	AUC
0.745	0.664	0.795	0.732

5.5 ROC Curves (Optional Visualization)

```
roc_train_tree <- roc(response = train$survived, predictor = tree_train_prob, quiet = TRUE)
roc_test_tree  <- roc(response = test$survived,  predictor = tree_test_prob,  quiet = TRUE)

plot(roc_train_tree, main = "Classification Tree - ROC Curves")
lines(roc_test_tree, lty = 2)
legend(
  "bottomright",
  legend = c(
    paste0("Train AUC = ", round(auc(roc_train_tree), 3)),
    paste0("Test AUC = ", round(auc(roc_test_tree), 3))
  ),
  lty = c(1, 2),
  bty = "n"
)
```



6. Challenger Model Comparison

Finally, we summarize the key performance metrics for both challenger models side-by-side on the **test set**.

```
challenger_comparison <- bind_rows(
  logit_test_eval$metrics  |> mutate(Model = "Logistic Regression", Split = "Test"),
  tree_test_eval$metrics   |> mutate(Model = "Classification Tree", Split = "Test")
) |>
  relocate(Model, Split)

kable(
  challenger_comparison,
  digits = 3,
  caption = "Challenger Models - Test Set Performance Comparison"
)
```

Table 5: Challenger Models – Test Set Performance Comparison

Model	Split	Accuracy	Sensitivity	Specificity	AUC
Logistic Regression	Test	0.752	0.672	0.800	0.794
Classification Tree	Test	0.745	0.664	0.795	0.732

Both challenger models—logistic regression and the classification tree—perform reasonably well on the test set, but the logistic regression model shows consistently stronger and more stable performance. It achieves a slightly higher accuracy (0.752 vs. 0.745) and a notably higher AUC (0.794 vs. 0.732), indicating superior ability to discriminate between survivors and non-survivors. Additionally, logistic regression exhibits less performance drop from training to testing, suggesting better generalization and lower overfitting.

While the classification tree offers intuitive, rule-based interpretability, this comes at the cost of reduced predictive power. Based on overall accuracy, sensitivity/specificity balance, and AUC, the logistic regression model is the preferred challenger model and should be selected as the stronger benchmark for comparison with the project's primary - champion model.