Report, Lab 2, Assignment 2

Max Goldbeck-Löwe

2025-12-03

Assignment 2. Decision Trees and Logistic Regression for Bank Marketing

1. Data Preperation

The data was divided into training (40%), validation (30%) and test (30%) sets and the variable "duration" was removed.

2. Decision Trees

Three different decision trees were fitted to the training data, out of which one had default settings, one had the smallest allowed node size equal to 7000 and the last had the minimum deviance equal to 0.0005. Below are the misclassification rates, rounded to four decimals, for both training and validation data for the three trees.

```
## Warning: package 'knitr' was built under R version 4.5.1
```

Table 1: Misclassification Rates on Training and Validation Data

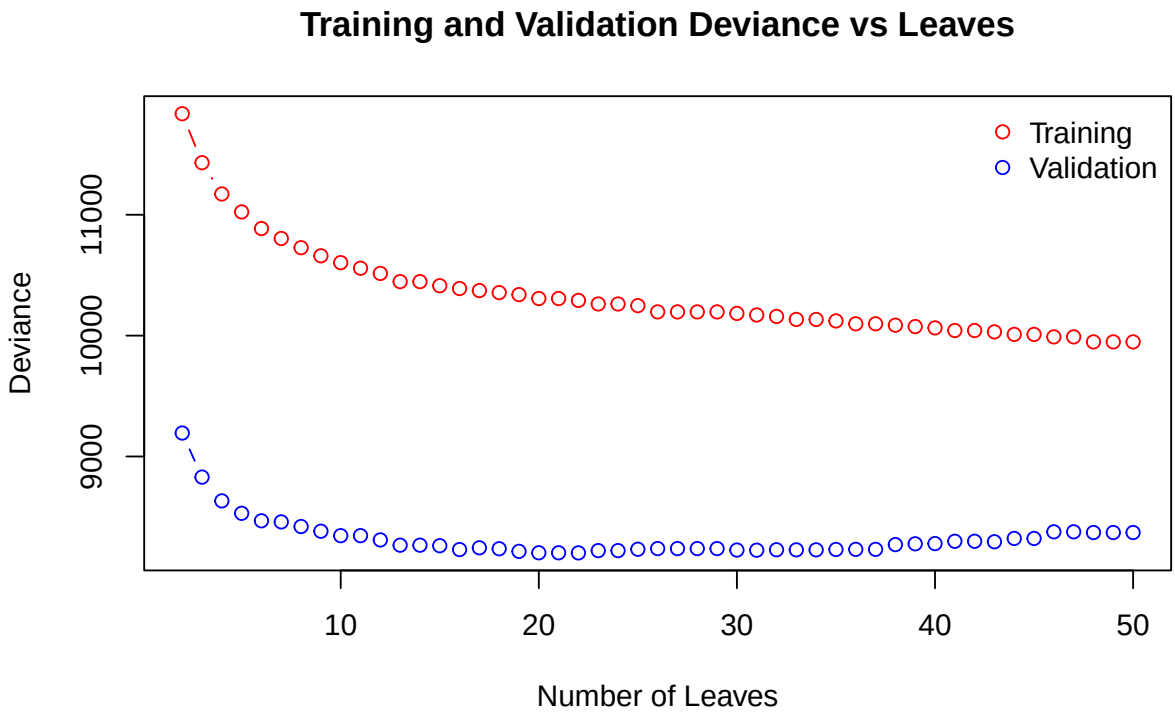
Trees	Training	Validation
Default Tree	0.1048	0.1093
Node Tree	0.1048	0.1093
Deviance Tree	0.0936	0.1118

While the tree with a minimum deviance fits the training data best out of the three, it performs slightly worse on the validation data compared to the other two, suggesting possible overfitting. Based on this, the default tree or the node tree, having identical misclassification rates, seem better choices.

When the minimum node size was set to 7000, it forced each terminal node to have more observations, reducing the number of splits. Hence, the tree for which this was done became slightly smaller and simpler than the default tree, with 5 terminal nodes as opposed to 6. When lowering the minimum deviance to 0.0005, however, more splits were allowed, resulting in a considerably larger and more complex tree with 122 nodes.

3. Optimal Tree Depth

Next, the optimal tree depth for the large (minimum deviance equal to 0.0005) tree was to be found. The development of deviance over up to 50 leaves was studied for both the training and validation sets, and the following graph was obtained.



As seen in the figure, the deviance for the training data continuously decreased as the number of leaves increased, though improvements became marginal at higher leaf counts. This indicates that the tree fit the training data better as it grew more complex. For the validation data, however, the deviance initially decreased, after which it leveled and even started to increase slightly. This demonstrates a bias-variance tradeoff, where more leaves at first resulted in a lower bias and thereby better fit, but as the tree became increasingly larger and more complex, while capturing even more of the training data, variance increased. The model started to overfit the training data and lost generalization ability, causing the increase in validation deviance.

The optimal number of leaves would be found where the validation deviance was at its lowest, approximately somewhere between 17 and 23, from reviewing the graph. To confirm and get an exact value, the minimum of the curve was evaluated and proved to occur when the number of leaves was equal to 22. The tree was then pruned to this optimal depth, and which variables were most important in decision making was examined. The variables "poutcome" (outcome of the previous marketing campaign) and "month" (last contact month of year) were the two most impactful, suggesting customers previous attitude towards bank term deposits, as well as in which month they were contacted, greatly affected their decision.

4. Confusion Matrix, Accuracy and F1 score

The confusion matrix for the test data, seen below, was estimated using the optimal tree obtained previously.

Table 2: Confusion Matrix 1

	Predicted: no	Predicted: yes
Observed: no	11872	107
Observed: yes	1371	214

The accuracy and F1 score were also calculated based on the confusion matrix. Accuracy was computed as

the proportion of correctly classified observations:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total observations}}$$

where TP and TN represent true positives and true negatives. Precision, recall, and F1 were derived as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Where FP and FN represent false positives and false negatives.

The model achieved an accuracy of **0.89** and an F1 score of **0.22**. While the accuracy appears high, this is largely due to the strong class imbalance in the dataset, as there were 39,922 observations labeled as "no" compared to only 5,289 labeled as "yes". In such cases, accuracy can be misleading because correctly predicting the majority class dominates the metric. The low F1 score reflects poor performance on the minority class ("yes"), as F1 combines precision and recall and is more sensitive to imbalanced data.

5. Loss Matrix

The test data was then classified again using the optimal tree, but this time with the addition of the loss matrix shown below.

Table 3: Loss Matrix

	Predicted: no	Predicted: yes
Observed: no	0	1
Observed: yes	5	0

To apply the loss matrix, the expected loss (EL) for each class was computed as:

$$\text{EL}(\text{predict yes}) = P(\text{no}) \cdot 1 + P(\text{yes}) \cdot 0, \quad \text{EL}(\text{predict no}) = P(\text{yes}) \cdot 5 + P(\text{no}) \cdot 0$$

The predicted class was chosen as the one with the lower expected loss. A new confusion matrix was then obtained.

Table 4: Confusion Matrix 2

	Predicted: no	Predicted: yes
Observed: no	11030	949
Observed: yes	771	814

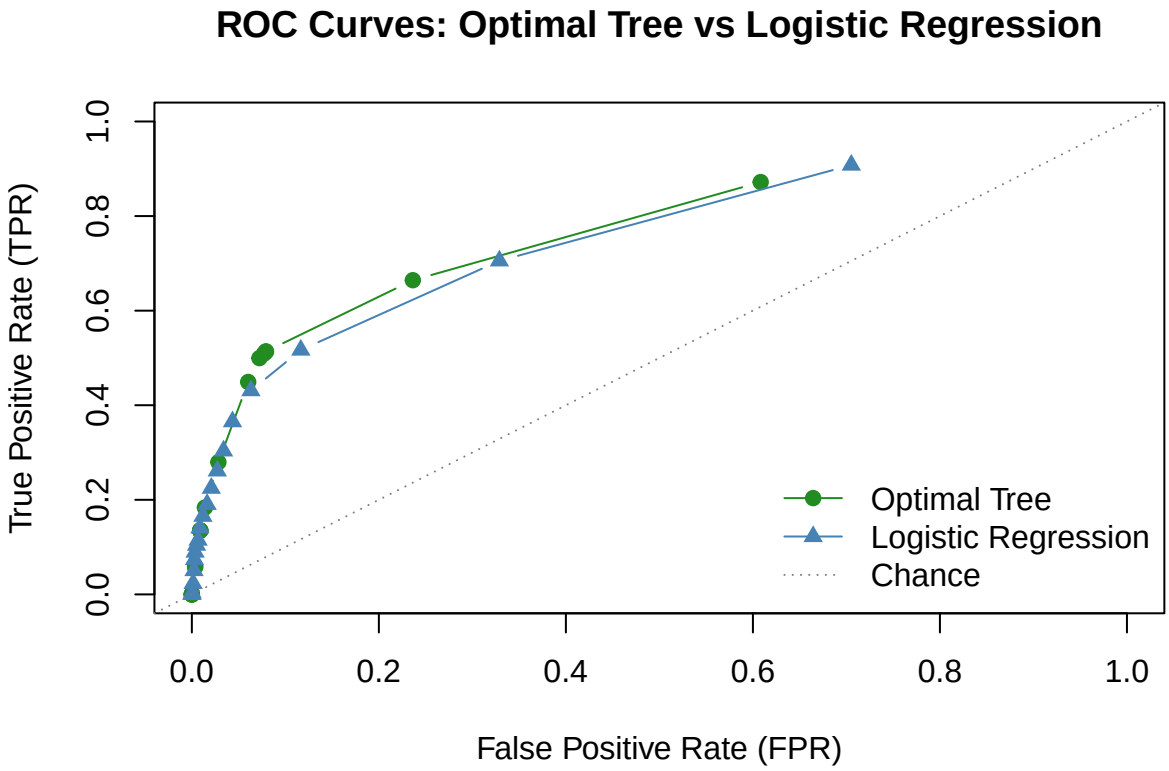
The confusion matrix shows a considerable increase in how much the model predicted "yes", which is explained by the relatively much larger penalization of misclassifying a "yes" as opposed to misclassifying a "no", imposed by the loss matrix. This resulted in a small decrease of the model accuracy, now being **0.87** (previously 0.89), but a quite large improvement of the F1 score to **0.49** (previously 0.22). While the model performed slightly less accurate predictions overall, it was significantly better when it came to the minority class ("yes").

6. ROC Curves

A logistic regression model was implemented on the training data, after which both that model and the optimal tree were used to classify the test data by the following principle:

$$\hat{Y} = \begin{cases} \text{yes}, & \text{if } p(Y = \text{'yes'} \mid X) > \pi \\ \text{no}, & \text{otherwise} \end{cases} \quad \text{where } \pi \in \{0.05, 0.10, 0.15, \ldots, 0.95\}$$

The TPR and FPR values where then computed for both models and the corresponding ROC curves are shown in the following plot.



The ROC curves for the optimal decision tree and logistic regression model both show performance well above the random baseline (represented by the dashed diagonal line). The optimal tree performed slightly better overall, as its curve lies marginally above the logistic regression, meaning it achieved higher true positive rates for the same false positive rates. However, because the dataset is highly imbalanced (many "no" and few "yes"), ROC curves can be misleading, as the false positive rate remains small even when the absolute number of false positives is large. In such cases, a precision–recall curve could be more informative because it focuses on the positive class, balancing precision (how many predicted "yes" are correct") and recall (how many of actual "yes" were captured), which is critical for campaign targeting. Essentially, you would want most of the people who you predicted would answer "yes" to actually do so, as contacting those who will not costs time and money, while simultaneously ensuring not to miss out on potential customers.

Assignment 4. Theory

2.

The important aspect when selecting mini-batches stated in the book (p.125) is that they should be randomly formed from shuffled data, to ensure they represent the overall dataset. For instance, if the dataset is sorted based on output class, a mini-batch might only include observations of class. One wants to avoid this, since such a mini-batch would "not give a good approximation of the gradient for the full dataset", which is what it is meant to do.

Appendix

```
#Assignment 2

#1.
data <- read.csv("bank-full.csv", sep =";")

data <- subset(data, select = -duration)

data[] <- lapply(data, function(x) {
  if (is.character(x)) as.factor(x) else x
})

n <- dim(data)[1]
set.seed(12345)
id <- sample(1:n, floor(n * 0.4))
train <- data[id, ]

id1 <- setdiff(1:n, id)
set.seed(12345)
id2 <- sample(id1, floor(n * 0.3))
valid <- data[id2, ]

id3 <- setdiff(id1, id2)
test <- data[id3, ]

#2.
library(tree)

#a)
tree_default <- tree(y ~ ., data = train)
summary(tree_default)

#b)
tree_nodesize <- tree(y ~ ., data = train, minsize = 7000)
summary(tree_nodesize)

#c)
tree_deviance <- tree(y~ ., data = train, mindev = 0.0005)
summary(tree_deviance)

Yfit_default <- predict(tree_default, newdata = valid, type = "class")
table(valid$y, Yfit_default)
```

```r
mean(Yfit_default != valid$y)

Yfit_nodesize <- predict(tree_nodesize, newdata = valid, type = "class")
table(valid$y, Yfit_nodesize)
mean(Yfit_nodesize != valid$y)

Yfit_deviance <- predict(tree_deviance, newdata = valid, type = "class")
table(valid$y, Yfit_deviance)
mean(Yfit_deviance != valid$y)

#3

trainScore <- rep(0, 50)
testScore <- rep(0, 50)

for (i in 2:50) {
  prunedTree <- prune.tree(tree_deviance, best = i)
  trainScore[i] <- deviance(prunedTree)
  pred <- predict(prunedTree, newdata = valid, type = "tree")
  testScore[i] <- deviance(pred)
}

plot(2:50, trainScore[2:50], type = "b", col = "red", ylim = c(7000, max(trainScore, testScore)),
     xlab = "Number of Leaves", ylab = "Deviance")
points(2:50, testScore[2:50], type = "b", col = "blue")

trainScore[2:50]
testScore[2:50]

optimal_leaves <- which.min(testScore[2:50]) + 1
optimal_leaves

opt_tree <- prune.tree(tree_deviance, best = 22)
summary(opt_tree)
plot(opt_tree)
text(opt_tree, pretty = 0)

#4

test_pred <- predict(opt_tree, newdata = test, type = "class")
conf_matrix <- table(test$y, test_pred)
conf_matrix

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

TP <- conf_matrix["yes", "yes"]
FP <- conf_matrix["no", "yes"]
FN <- conf_matrix["yes", "no"]

precision <- TP / (TP + FP)
recall <- TP / (TP + FN)
F1 <- 2 * precision * recall / (precision + recall)
```

```
F1

sum(data$y == "yes")
sum(data$y == "no")

#5


probs <- predict(opt_tree, newdata = test, type = "vector")

loss_matrix <- matrix(c(0, 1, 5, 0), nrow = 2, byrow = TRUE)
rownames(loss_matrix) <- colnames(loss_matrix) <- c("no", "yes")

expected_loss_yes <- probs[, "yes"] * loss_matrix["yes", "yes"] + probs[, "no"] * loss_matrix["no", "yes
expected_loss_no  <- probs[, "yes"] * loss_matrix["yes", "no"]  + probs[, "no"] * loss_matrix["no", "no

test_pred_loss <- ifelse(expected_loss_yes < expected_loss_no, "yes", "no")

conf_matrix_loss <- table(test$y, test_pred_loss)
conf_matrix_loss

accuracy_loss <- sum(diag(conf_matrix_loss)) / sum(conf_matrix_loss)
accuracy_loss

TP_loss <- conf_matrix_loss["yes", "yes"]
FP_loss <- conf_matrix_loss["no", "yes"]
FN_loss <- conf_matrix_loss["yes", "no"]

precision_loss <- TP_loss / (TP_loss + FP_loss)
recall_loss <- TP_loss / (TP_loss + FN_loss)
F1_loss <- 2 * precision_loss * recall_loss / (precision_loss + recall_loss)
F1_loss

#6

logit_mod <- glm(y ~ ., data = train, family = binomial)

p_logit <- predict(logit_mod, newdata = test, type = "response")

p_tree <- probs[, "yes"]

compute_tpr_fpr <- function(p, y_true, thresholds) {
  res <- lapply(thresholds, function(pi) {
    pred <- ifelse(p > pi, "yes", "no")
    cm <- table(y_true, factor(pred, levels = c("no", "yes")))
    TP <- cm["yes", "yes"]; FN <- cm["yes", "no"]
    FP <- cm["no",  "yes"]; TN <- cm["no",  "no"]
    TPR <- ifelse((TP + FN) > 0, TP / (TP + FN), NA)
    FPR <- ifelse((FP + TN) > 0, FP / (FP + TN), NA)
    data.frame(threshold = pi, TPR = TPR, FPR = FPR)
  })
  do.call(rbind, res)
}
```

```r
pis <- seq(0.05, 0.95, by = 0.05)

roc_tree  <- compute_tpr_fpr(p_tree,  test$y, pis)
roc_logit <- compute_tpr_fpr(p_logit, test$y, pis)

roc_tree
roc_logit

plot(roc_tree$FPR, roc_tree$TPR, type = "b", pch = 19, col = "forestgreen",
     xlim = c(0, 1), ylim = c(0, 1),
     xlab = "FPR",
     ylab = "TPR",
     main = "ROC Curves")
lines(roc_logit$FPR, roc_logit$TPR, type = "b", pch = 17, col = "steelblue")
abline(0, 1, lty = 3, col = "grey50")
```