ADM\_Assignment02

Chandima Attanayake

2025-04-06

**Part A**

*QA1. Key idea behind bagging*

Bagging is an ensemble method that reduces variance by training multiple models on different bootstrapped samples of the original or training dataset and then averaging (regression) or voting (classification) their prediction.

It is particularly effective against high variance (overfitting) because it smooths out the predictions. Therefore, the answer is yes, bagging is particularly effective at reducing variance.

However, bagging is not effective in reducing high bias (underfitting) because all the individual models are of the same type and have similar bias. Therefore, the answer is no

*QA2. Computational Efficiency of Bagging vs Boosting*

Bagging trains all models in parallel as each model is independent of the others. Also, boosting trains models sequentially, where each model learns from the errors of the previous one. Additionally, bagging does not require adaptive reweighting of samples, unlike boosting.

Therefore, the bagging is more efficient than boosting.

*QA3. Ensemble of Weak Decision Trees*

No, combining very similar or weak models will likely not improve performance as If individual models perform no better than random guessing, they are too weak (high bias). Also, Since the models are similar, ensembling them will not correct errors effectively. Addition to that, it is required a stronger base model (deeper trees) or a different approach like boosting, which focuses on correcting errors. Therefore, it seems that combining these trees will not improve performance.

*QA4.Information Gain for Splitting Based on the size attribute*

Step 01. Computer Parent Entropy (Before Split)

• Total records = 16 • Edible (+) = 8, Non-Edible (–) = 8 • Entropy(S) = −(P(+)log₂P(+) + P(−)log₂P(−)) = −(0.5 × log₂(0.5) + 0.5 × log₂(0.5)) = −(0.5 × (−1) + 0.5 × (−1)) = 1

Step 02. Split by “Size” and Compute Child Entropies

Size Edible (+) Non-Edible (–) Total Small 6 2 8 Large 2 6 8

Entropy(Small) = −( (6/8)log₂(6/8) + (2/8)log₂(2/8) ) = −(0.75 × (−0.415) + 0.25 × (−2)) = 0.811

Entropy(Large) = −( (2/8)log₂(2/8) + (6/8)log₂(6/8) ) = −(0.25 × (−2) + 0.75 × (−0.415)) = 0.811

Step 03. Weighted Average Entropy After Split

Weighted Entropy = ((8/16)×0.811)+((8/16)×0.811) = 0.811

Step 04. Information Gain

IG(Size) = Entropy(S)−Weighted Entropy=1−0.811 = 0.189

Accordingly, the Information Gain for splitting on Size is 0.189.

*QA5.Importance of m (mtry) in Random Forest*

The optimal m, controls the number of features considered at each split (default: √p for classification, p/3 for regression).

A small m value increases diversity among trees, helping reduce correlation, which often improves generalization. A large m increases similarity among trees, reducing diversity, and might not reduce variance effectively.

Implication of the m size can be seen as below.

Implications

Too Small

• Trees become more independent (good for diversity). • But each tree is weaker (may underfit).

Too Large

• Trees become too similar (lose diversity, like bagging without feature randomness). • Higher correlation between trees → less variance reduction.

The ideal situation is tuning m via cross-validation (caret does this automatically) or a moderate m (e.g., between 2 and √p) usually works best.

**Part B**

*QB1: Building a Decision Tree Regression Model*

# Load required libraries  
library(ISLR)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-8

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

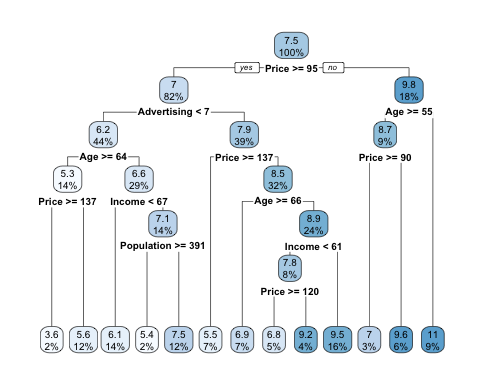
library(rpart) # For decision trees  
library(rpart.plot) # For visualizing decision trees

## Warning: package 'rpart.plot' was built under R version 4.4.1

install.packages("randomForest")

##   
## The downloaded binary packages are in  
## /var/folders/b1/tmzk08b13tx9r4sqw3lzkmmr0000gn/T//RtmpFydGiB/downloaded\_packages

# Filter the dataset  
Carseats\_Filtered <- Carseats %>%  
 select(Sales, Price, Advertising, Population, Age, Income, Education)  
  
# Build the decision tree model  
set.seed(123) # For reproducibility  
tree\_model <- rpart(Sales ~ ., data = Carseats\_Filtered)  
  
# Visualize the tree  
rpart.plot(tree\_model)



*QB2: Predicting Sales for a Specific Record*

# Create the new data point  
new\_data <- data.frame(  
 Sales = 9,  
 Price = 6.54,  
 Population = 124,  
 Advertising = 0,  
 Age = 76,  
 Income = 110,  
 Education = 10  
)  
  
# Predict using the decision tree model  
predicted\_sales <- predict(tree\_model, newdata = new\_data)  
predicted\_sales

## 1   
## 9.58625

The predicting sales value ould be 9.58625.

*QB3: Training a Random Forest with caret*

# Set seed for reproducibility  
set.seed(123)  
  
# Set up training control  
trControl <- trainControl(method = "cv", number = 5) # 5-fold CV  
  
# Train the random forest model  
rf\_model <- train(Sales ~ .,   
 data = Carseats\_Filtered,  
 method = "rf",  
 trControl = trControl)  
  
# View the results  
print(rf\_model)

## Random Forest   
##   
## 400 samples  
## 6 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 320, 321, 319, 320, 320   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared MAE   
## 2 2.406539 0.2838955 1.926998  
## 4 2.405609 0.2874877 1.916925  
## 6 2.415585 0.2834264 1.924429  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 4.

*QB4: Custom Grid Search with Specific mtry Values*

# Define the tuning grid  
tuneGrid <- expand.grid(mtry = c(2, 3, 5))  
  
# Set up training control with repeats  
trControl <- trainControl(method = "repeatedcv",   
 number = 5,   
 repeats = 3)  
  
# Train the model with custom grid  
rf\_model\_custom <- train(Sales ~ .,   
 data = Carseats\_Filtered,  
 method = "rf",  
 tuneGrid = tuneGrid,  
 trControl = trControl)  
  
# View the results  
print(rf\_model\_custom)

## Random Forest   
##   
## 400 samples  
## 6 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold, repeated 3 times)   
## Summary of sample sizes: 319, 320, 320, 321, 320, 320, ...   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared MAE   
## 2 2.360409 0.3086974 1.887810  
## 3 2.357353 0.3091664 1.884653  
## 5 2.359763 0.3103808 1.885362  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 3.