Assignment 3 Walkthrough

Warren Mansur

Reminder: Purpose of Walkthrough

- The goal of this walkthrough is to help you successfully understand and complete Assignment 3.

Reminder: Purpose of Walkthrough

- The goal of this walkthrough is to help you successfully understand and complete Assignment 3.
- I briefly recap important concepts relevant to the assignment, then show you relevant code examples.

Reminder: Purpose of Walkthrough

- The goal of this walkthrough is to help you successfully understand and complete Assignment 3.
- I briefly recap important concepts relevant to the assignment, then show you relevant code examples.
- These code examples are not shown elsewhere (i.e. not in the instructor's live class, shared example code, or online modules).

Entropy: What it Is

- Entropy is a measure of uncertainty.

Entropy: What it Is

- Entropy is a measure of uncertainty.
  - We're focusing on two possible outcomes, like Yes or No.

Entropy: What it Is

- Entropy is a measure of uncertainty.
  - We're focusing on two possible outcomes, like Yes or No.
- When the outcome is fully predictable, entropy is **0 bits**.

Entropy: What it Is

- Entropy is a measure of uncertainty.
  - We're focusing on two possible outcomes, like Yes or No.
- When the outcome is fully predictable, entropy is **0 bits**.
  - No information is needed; the outcome is already known in advance.

Entropy: What it Is

- Entropy is a measure of uncertainty.
  - We're focusing on two possible outcomes, like Yes or No.
- When the outcome is fully predictable, entropy is **0 bits**.
  - No information is needed; the outcome is already known in advance.
- When the outcome is completely unpredictable, like a fair coin flip, entropy is **1 bit**.

Entropy: What it Is

- Entropy is a measure of uncertainty.
  - We're focusing on two possible outcomes, like Yes or No.
- When the outcome is fully predictable, entropy is **0 bits**.
  - No information is needed; the outcome is already known in advance.
- When the outcome is completely unpredictable, like a fair coin flip, entropy is **1 bit**.
  - One yes/no bit is required to specify whether the flip was heads or tails.

Entropy: What it Is

- Entropy is a measure of uncertainty.
  - We're focusing on two possible outcomes, like Yes or No.
- When the outcome is fully predictable, entropy is **0 bits**.
  - No information is needed; the outcome is already known in advance.
- When the outcome is completely unpredictable, like a fair coin flip, entropy is **1 bit**.
  - One yes/no bit is required to specify whether the flip was heads or tails.
- When one is more likely than the other (say 75%/25%), entropy is about **0.81 bit**.

Entropy: What it Is

- Entropy is a measure of uncertainty.
  - We're focusing on two possible outcomes, like Yes or No.
- When the outcome is fully predictable, entropy is **0 bits**.
  - No information is needed; the outcome is already known in advance.
- When the outcome is completely unpredictable, like a fair coin flip, entropy is **1 bit**.
  - One yes/no bit is required to specify whether the flip was heads or tails.
- When one is more likely than the other (say 75%/25%), entropy is about **0.81 bit**.
  - Because the result is partly predictable, you need less than one bit on average to encode it.

6-Sided Dice Roll Example Questions

Step	Question	What the answers mean
1	Is the number greater than 3?	**Yes →** 4, 5, 6 **No →** 1, 2, 3
2	Is the number even? (same question in either branch)	▪ First answer **No**, this **Yes** → 2 (done) ▪ First answer **Yes**, this **No** → 5 (done) ▪ Otherwise two possibilities remain — go to **Step 3**
3	Tie-breaker (only when two choices remain)	1–3 branch: **Is the number 1?** – Yes → 1 – No → 3 4–6 branch: **Is the number 4?** – Yes → 4 – No → 6

Entropy for 6-Sided Dice Roll

Roll	Q1 (>3?)	Q2 (even?)	Q3	Questions used
1	No	No	"Is it 1?" → Yes	3
2	No	**Yes**	—	2
3	No	No	"Is it 1?" → No	3
4	**Yes**	**Yes**	"Is it 4?" → Yes	3
5	Yes	**No**	—	2
6	Yes	Yes	"Is it 4?" → No	3

- Two faces (2 and 5) need just two questions; the other four faces each need three.

Entropy for 6-Sided Dice Roll

Roll	Q1 (>3?)	Q2 (even?)	Q3	Questions used
1	No	No	"Is it 1?" → Yes	3
2	No	**Yes**	—	2
3	No	No	"Is it 1?" → No	3
4	**Yes**	**Yes**	"Is it 4?" → Yes	3
5	Yes	**No**	—	2
6	Yes	Yes	"Is it 4?" → No	3

- Two faces (2 and 5) need just two questions; the other four faces each need three.
- That totals $(2 + 4 \times 3 + 2)/6 = 2.67$ questions on average, the best you can do for a six-sided die.

Understanding Students Pass/Fail Entropy Encoding

- Imagine **75** students pass, and **25** students fail, and that's all you know.

Pair (two students)	Probability	Code word	Length
Pass Pass	(0.75^2 = 0.5625)	0	1 bit
Pass Fail	(0.75 ×0.25 = 0.1875)	10	2 bits
Fail Pass	(0.25 ×0.75 = 0.1875)	110	3 bits
Fail Fail	(0.25^2 = 0.0625)	111	3 bits

Understanding Students Pass/Fail Entropy Encoding

- Imagine **75** students pass, and **25** students fail, and that's all you know.
- Using Pass $= 0$ and Fail $= 1$ is wasteful. You pay 1 full bit for every student.

Pair (two students)	Probability	Code word	Length
Pass Pass	$(0.75\hat{}2 = 0.5625)$	0	1 bit
Pass Fail	$(0.75 \times 0.25 = 0.1875)$	10	2 bits
Fail Pass	$(0.25 \times 0.75 = 0.1875)$	110	3 bits
Fail Fail	$(0.25\hat{}2 = 0.0625)$	111	3 bits

Understanding Students Pass/Fail Entropy Encoding

- Imagine **75** students pass, and **25** students fail, and that's all you know.
- Using Pass $= 0$ and Fail $= 1$ is wasteful. You pay 1 full bit for every student.

Pair (two students)	Probability	Code word	Length
Pass Pass	(0.75^2 = 0.5625)	0	1 bit
Pass Fail	(0.75 ×0.25 = 0.1875)	10	2 bits
Fail Pass	(0.25 ×0.75 = 0.1875)	110	3 bits
Fail Fail	(0.25^2 = 0.0625)	111	3 bits

- Average for the pair: $0.5625 * 1 + 0.1875 * 2 + 0.1875 * 3 + 0.0625 * 3 \approx 1.62$ bits.

Understanding Students Pass/Fail Entropy Encoding

- Imagine **75** students pass, and **25** students fail, and that's all you know.
- Using Pass $= 0$ and Fail $= 1$ is wasteful. You pay 1 full bit for every student.

Pair (two students)	Probability	Code word	Length
Pass Pass	(0.75^2 = 0.5625)	0	1 bit
Pass Fail	(0.75 ×0.25 = 0.1875)	10	2 bits
Fail Pass	(0.25 ×0.75 = 0.1875)	110	3 bits
Fail Fail	(0.25^2 = 0.0625)	111	3 bits

- Average for the pair: $0.5625 * 1 + 0.1875 * 2 + 0.1875 * 3 + 0.0625 * 3 \approx 1.62$ bits.
  - Dividing by 2 students=0.81 bit each.

Understanding Students Pass/Fail Entropy Encoding

- Imagine **75** students pass, and **25** students fail, and that's all you know.
- Using Pass = 0 and Fail = 1 is wasteful. You pay 1 full bit for every student.

Pair (two students)	Probability	Code word	Length
Pass Pass	(0.75^2 = 0.5625)	0	1 bit
Pass Fail	(0.75 ×0.25 = 0.1875)	10	2 bits
Fail Pass	(0.25 ×0.75 = 0.1875)	110	3 bits
Fail Fail	(0.25^2 = 0.0625)	111	3 bits

- Average for the pair: $0.5625 * 1 + 0.1875 * 2 + 0.1875 * 3 + 0.0625 * 3 \approx 1.62$ bits.
  - Dividing by 2 students=0.81 bit each.
- Key insight: With variable-length code words and encoding blocks, the cheap codes dominate, driving the cost down to the theoretical minimum.

Information Gain (Requires a Feature)

- Now imagine you know whether the student submitted all homework (extra feature).

Pile	Size	Pass (“Yes”)	Fail (“No”)
Pile A – turned everything in	60 students	54	6
Pile B – missed 1 assignment	40 students	21	19

Information Gain (Requires a Feature)

- Now imagine you know whether the student submitted all homework (extra feature).

Pile	Size	Pass (“Yes”)	Fail (“No”)
Pile A – turned everything in	60 students	54	6
Pile B – missed 1 assignment	40 students	21	19

- **Pile A entropy**: $-\left(0.9 \log_2 0.9 + 0.1 \log_2 0.1\right) \approx 0.47$ bit

Information Gain (Requires a Feature)

- Now imagine you know whether the student submitted all homework (extra feature).

Pile	Size	Pass (“Yes”)	Fail (“No”)
Pile A – turned everything in	60 students	54	6
Pile B – missed 1 assignment	40 students	21	19

- **Pile A entropy**: $-(0.9 \log_2 0.9 + 0.1 \log_2 0.1) \approx 0.47$ bit
- **Pile B entropy**: $-(0.6 \log_2 0.6 + 0.4 \log_2 0.4) \approx 0.97$ bit

Information Gain (Requires a Feature)

- Now imagine you know whether the student submitted all homework (extra feature).

Pile	Size	Pass (“Yes”)	Fail (“No”)
Pile A – turned everything in	60 students	54	6
Pile B – missed 1 assignment	40 students	21	19

- **Pile A entropy**: $-(0.9 \log_2 0.9 + 0.1 \log_2 0.1) \approx 0.47$ bit
- **Pile B entropy**: $-(0.6 \log_2 0.6 + 0.4 \log_2 0.4) \approx 0.97$ bit
- **Combine them, honoring their sizes**: $0.60 \times 0.47 + 0.40 \times 0.97 \approx 0.60$ bit

Information Gain (Requires a Feature)

- Now imagine you know whether the student submitted all homework (extra feature).

Pile	Size	Pass (“Yes”)	Fail (“No”)
Pile A – turned everything in	60 students	54	6
Pile B – missed 1 assignment	40 students	21	19

- **Pile A entropy**: $-(0.9 \log_2 0.9 + 0.1 \log_2 0.1) \approx 0.47$ bit
- **Pile B entropy**: $-(0.6 \log_2 0.6 + 0.4 \log_2 0.4) \approx 0.97$ bit
- **Combine them, honoring their sizes**: $0.60 \times 0.47 + 0.40 \times 0.97 \approx 0.60$ bit
- **See the payoff**: Information Gain=before-after= 0.81 bit $- 0.60$ bit $= 0.21$ bit

Information Gain (Requires a Feature)

- Now imagine you know whether the student submitted all homework (extra feature).

Pile	Size	Pass (“Yes”)	Fail (“No”)
Pile A – turned everything in	60 students	54	6
Pile B – missed 1 assignment	40 students	21	19

- **Pile A entropy**: $-(0.9 \log_2 0.9 + 0.1 \log_2 0.1) \approx 0.47$ bit
- **Pile B entropy**: $-(0.6 \log_2 0.6 + 0.4 \log_2 0.4) \approx 0.97$ bit
- **Combine them, honoring their sizes**: $0.60 \times 0.47 + 0.40 \times 0.97 \approx 0.60$ bit
- **See the payoff**: Information Gain=before-after$= 0.81$ bit $- 0.60$ bit $= 0.21$ bit
- With no features, it takes 0.81 bit to decide, yet the one feature reduces it to 0.60 bit, 0.21 less.

Finding Best Information Gain - Part 1

```r
df <- read.csv("loan_approval.csv",            # CSV in your working directory
               stringsAsFactors = TRUE)        # keep text columns as factors

# Entropy Function
entropy <- function(vec) {                     # vec = factor of class labels
  probs <- prop.table(table(vec))              # convert counts to probabilities
  -sum(probs * log2(probs))                    # apply −Σ p log₂ p
}

# Information Gain Function
info_gain <- function(data, target, attribute) {           # data frame, names as strings
  ent_parent <- entropy(data[[target]])                    # entropy of the whole set
  splits <- split(data[[target]], data[[attribute]])       # class labels partitioned by attribute value
  weights <- sapply(splits, length) / nrow(data)           # subset sizes as fractions
  ent_children <- mapply(function(v, w) w * entropy(v),    # weighted entropies of subsets
                         splits, weights, SIMPLIFY = TRUE)
  gain <- ent_parent - sum(ent_children)                   # parent − children = information gain
  return(gain)                                             # send IG back to caller
}
```

Finding Best Information Gain - Part 2

```r
# ---------- 4.  Columns to analyze ------------------------
target_col <- "Approved"                                # binary class column
candidate_attrs <- c("IncomeBracket", "HomeOwnership")     # predictors to compare

# ---------- 5.  Compute IG for each predictor -------------
ig_results <- sapply(candidate_attrs,                      # loop over predictors
                     function(attr) info_gain(df,          # compute IG on each
                                              target_col,
                                              attr))

# ---------- 6.  Show results ------------------------------
print(round(ig_results, 4))                             # IG values (4-dec precision)
best_attr <- names(which.max(ig_results))                  # attribute with max IG
cat("\nBest root split by information gain:", best_attr, "\n")  # report winner
```

Finding Best Information Gain - Results

```
IncomeBracket HomeOwnership
       0.0065        0.0850


Best root split by information gain: HomeOwnership
```

Using Naive Bayes with Healthy Diet Dataset 1-2

```r
library(tidyverse)   # pipes + data wrangling
library(rsample)     # initial_split(), training(), testing()
library(e1071)       # naiveBayes()
library(caret)       # confusionMatrix()

## 1  Load, convert "?" to NA, tidy types -------------------------------------
df_raw <- read.csv(
  "healthy_diet.csv",
  stringsAsFactors = FALSE,
  na.strings      = c("?", "")      # <- treats "?" and blanks as NA
)

# Factor vs. numeric columns (edit if your schema differs)
factor_cols <- c(
  "gender", "ethnicity", "food_allergy_history",
  "family_healthy_eater", "country_of_res",
  "used_food_tracker_before", "relation"
)
df_raw[factor_cols] <- lapply(df_raw[factor_cols], factor)
df_raw$Healthy_Diet <- factor(df_raw$Healthy_Diet, levels = c("NO", "YES"))

## 2  Handle missing data -----------------------------------------------------
# Simple strategy from lectures/R file: drop incomplete rows
df <- na.omit(df_raw)
```

Using Naive Bayes with Healthy Diet Dataset 3-5

```r
## 3  Train / test split ------------------------------------------------------
set.seed(699)                                # reproducible grading
split <- initial_split(df, prop = 0.66, strata = Healthy_Diet)
train <- training(split)
test  <- testing(split)

## 4  Fit Naïve Bayes ---------------------------------------------------------
model_nb <- naiveBayes(Healthy_Diet ~ ., data = train, laplace = 1)

## 5  Evaluate ----------------------------------------------------------------
pred <- predict(model_nb, newdata = test, type = "class")

perf <- confusionMatrix(
  data      = pred,
  reference = test$Healthy_Diet,
  positive  = "YES"
)

print(perf)       # accuracy, recall, F1, etc.
################################################################################
```

Using Naive Bayes with Healthy Diet Dataset Results

```
Confusion Matrix and Statistics

          Reference
Prediction NO YES
       NO  79  26
       YES 20  57

               Accuracy : 0.7473
                 95% CI : (0.6776, 0.8086)
    No Information Rate : 0.544
    P-Value [Acc > NIR] : 1.216e-08

                  Kappa : 0.4876

 Mcnemar's Test P-Value : 0.461

            Sensitivity : 0.6867
            Specificity : 0.7980
         Pos Pred Value : 0.7403
         Neg Pred Value : 0.7524
             Prevalence : 0.4560
         Detection Rate : 0.3132
   Detection Prevalence : 0.4231
      Balanced Accuracy : 0.7424

       'Positive' Class : YES
```

Naive Bayes Interpretation

- The overall accuracy of 0.75 is well above the 0.54 obtained by only predicting the majority class.

Naive Bayes Interpretation

- The overall accuracy of 0.75 is well above the 0.54 obtained by only predicting the majority class.
- The sensitivity (recall for the YES class) is 0.69 means the model finds about two-thirds of "healthy-diet" cases, but misses 31% of them. If catching every positive is critical, this model may not be suitable.

Naive Bayes Interpretation

- The overall accuracy of 0.75 is well above the 0.54 obtained by only predicting the majority class.
- The sensitivity (recall for the YES class) is 0.69 means the model finds about two-thirds of "healthy-diet" cases, but misses 31% of them. If catching every positive is critical, this model may not be suitable.
- The specificity of .80 means, out of the 99 people who do not follow a healthy diet, 79 were correctly categorized. This is not bad, but if catching every negative is critical, this model may not be suitable.

Naive Bayes Interpretation

- The overall accuracy of 0.75 is well above the 0.54 obtained by only predicting the majority class.
- The sensitivity (recall for the YES class) is 0.69 means the model finds about two-thirds of "healthy-diet" cases, but misses 31% of them. If catching every positive is critical, this model may not be suitable.
- The specificity of .80 means, out of the 99 people who do not follow a healthy diet, 79 were correctly categorized. This is not bad, but if catching every negative is critical, this model may not be suitable.
- The precision (pos pred value) of 0.74 means when the model predicts YES, it is right about 3/4 of the time. It could leave too many false positives if the model is used for follow-up actions on this class.