

W25_HW3

Cameron Bentz

March 24, 2025

Problem 1: Age Data Preprocessing

Suppose that we have age data including the following numbers in sorted order. Then answer the questions below.

```
age <- c(13, 15, 16, 16, 19, 20, 20, 21, 22, 22,
        25, 25, 25, 25, 30, 33, 33, 35, 35, 35,
        35, 36, 40, 45, 46, 52, 70)
```

(a) Smoothing by bin means (bin depth = 3)

```
depth <- 3
nbins <- floor(length(age) / depth)

smoothed <- c() # create empty vector
for (i in 0:(nbins - 1)) { # loop through bins of data
  start <- i * depth + 1
  end <- start + depth - 1
  bin <- age[start:end] # add age data to bin
  bin_mean <- mean(bin) # calculate mean
  smoothed <- c(smoothed, rep(bin_mean, depth)) # populate bin with mean values
}

# add remaining values
if (length(age) > nbins * depth) {
  remaining <- age[(nbins * depth + 1):length(age)]
  smoothed <- c(smoothed, rep(mean(remaining), length(remaining)))
}

smoothed_result <- data.frame(Original = age, Smoothed = smoothed)
smoothed_result
```

```
##      Original Smoothed
## 1         13 14.66667
## 2         15 14.66667
## 3         16 14.66667
## 4         16 18.33333
```

```
## 5      19 18.33333
## 6      20 18.33333
## 7      20 21.00000
## 8      21 21.00000
## 9      22 21.00000
## 10     22 24.00000
## 11     25 24.00000
## 12     25 24.00000
## 13     25 26.66667
## 14     25 26.66667
## 15     30 26.66667
## 16     33 33.66667
## 17     33 33.66667
## 18     35 33.66667
## 19     35 35.00000
## 20     35 35.00000
## 21     35 35.00000
## 22     36 40.33333
## 23     40 40.33333
## 24     45 40.33333
## 25     46 56.00000
## 26     52 56.00000
## 27     70 56.00000
```

Comment: Smoothing by bin means reduces variance and helps reveal patterns, but can distort outliers.

(b) Outlier detection using IQR

```
Q1 <- quantile(age, 0.25)
Q3 <- quantile(age, 0.75)
IQR_val <- IQR(age)

lower_bound <- Q1 - (1.5 * IQR_val)
upper_bound <- Q3 + (1.5 * IQR_val)

outliers <- age[age < lower_bound | age > upper_bound]
cat("The outliers for the data set are:", outliers, "\n")
```

```
## The outliers for the data set are: 70
```

(c) Min-max normalization of age 35 to [0.0, 1.0]

```
minmax <- (35 - min(age)) / (max(age) - min(age))
cat("The min-max normalization of age 35 to [0.0, 1.0] is:", minmax, "\n")
```

```
## The min-max normalization of age 35 to [0.0, 1.0] is: 0.3859649
```

(d) Z-score normalization of age 35

```
mean_age <- mean(age)
sd_age <- sd(age)
z_score <- (35 - mean_age) / sd_age
cat("The z-score normalization of age 35 is:", z_score, "\n")
```

```
## The z-score normalization of age 35 is: 0.3891971
```

(e) Decimal scaling normalization of age 35

```
k <- nchar(as.character(max(abs(age))))
decimal_scaled <- 35 / (10^k)
cat("The decimal scaling normalization of age 35 is:", decimal_scaled, "\n")
```

```
## The decimal scaling normalization of age 35 is: 0.35
```

Problem 2: Min-Max Normalization Function

```
minmax_normalize <- function(a, min_new, max_new) {
  min_old <- min(a)
  max_old <- max(a)
  scaled <- (a - min_old) / (max_old - min_old)
  scaled * (max_new - min_new) + min_new
}

# Test usage

cat("The min max normalization function with values 10, 20 produces:\n")
```

```
## The min max normalization function with values 10, 20 produces:
```

```
cat(paste(minmax_normalize(age, 10, 20), collapse = "\n"), "\n\n")
```

```
## 10
## 10.3508771929825
## 10.5263157894737
## 10.5263157894737
## 11.0526315789474
## 11.2280701754386
## 11.2280701754386
## 11.4035087719298
## 11.5789473684211
## 11.5789473684211
## 12.1052631578947
## 12.1052631578947
```

```
## 12.1052631578947
## 12.1052631578947
## 12.9824561403509
## 13.5087719298246
## 13.5087719298246
## 13.859649122807
## 13.859649122807
## 13.859649122807
## 13.859649122807
## 14.0350877192982
## 14.7368421052632
## 15.6140350877193
## 15.7894736842105
## 16.8421052631579
## 20
```

```
cat("The min max normalization function with values 0.0 , 1.0 produces:\n")
```

```
## The min max normalization function with values 0.0 , 1.0 produces:
```

```
cat(paste(minmax_normalize(age, 0, 1), collapse = "\n"))
```

```
## 0
## 0.0350877192982456
## 0.0526315789473684
## 0.0526315789473684
## 0.105263157894737
## 0.12280701754386
## 0.12280701754386
## 0.140350877192982
## 0.157894736842105
## 0.157894736842105
## 0.210526315789474
## 0.210526315789474
## 0.210526315789474
## 0.210526315789474
## 0.298245614035088
## 0.350877192982456
## 0.350877192982456
## 0.385964912280702
## 0.385964912280702
## 0.385964912280702
## 0.385964912280702
## 0.403508771929825
## 0.473684210526316
## 0.56140350877193
## 0.578947368421053
## 0.684210526315789
## 1
```

Problem 3: Information Gain for Decision Tree

The formula for information gain is given by $\text{InfoGain}(T, A) = \text{Entropy}(T) - \sum_{v \in \text{Values}(A)} \frac{|T_v|}{|T|} \cdot \text{Entropy}(T_v)$, where T is the full dataset and T_v is a subset corresponding to value v of attribute A .

```
data <- data.frame(
  department = c("sales", "sales", "sales", "systems", "systems", "systems", "systems",
                 "marketing", "marketing", "secretary", "secretary"),
  age = c("31_35", "26_30", "31_35", "21_25", "31_35", "26_30",
          "41_45", "36_40", "31_35", "46_50", "26_30"),
  salary = c("46K_50K", "26K_30K", "31K_35K", "46K_50K", "66K_70K", "46K_50K",
             "66K_70K", "46K_50K", "41K_45K", "36K_40K", "26K_30K"),
  status = c("senior", "junior", "junior", "junior", "senior", "junior",
             "senior", "senior", "junior", "senior", "junior"),
  count = c(30, 40, 40, 20, 5, 3, 3, 10, 4, 4, 6)
)

#print(data)

entropy <- function(p) {
  p <- p[p > 0]
  -sum(p * log2(p))
}

# Step 1: calculate overall dataset entropy
status_total <- aggregate(count ~ status, data = data, sum)
total_count <- sum(status_total$count)
p_class <- status_total$count / total_count
entropy_total <- entropy(p_class)
cat("The entropy of the dataset is:", entropy_total, "\n")

## The entropy of the dataset is: 0.8990308

# Step 2: calculate informtion for attributes
info_gain <- function(df, attr) {
  vals <- unique(df[[attr]]) # get unique attribute values
  weighted_entropy <- 0

  for (val in vals) {
    subset <- df[df[[attr]] == val, ]
    subset_total <- sum(subset$count) # count number of records in subset
    junior_count <- sum(subset$count[subset$status == "junior"])
    senior_count <- sum(subset$count[subset$status == "senior"])

    p_junior <- junior_count / subset_total #probability (junior)
    p_senior <- senior_count / subset_total #probability (senior)

    subset_entropy <- entropy(c(p_junior, p_senior))

    weighted_entropy <- weighted_entropy + (subset_total / total_count) * subset_entropy
  }

  # info gain = total entropy - weighted subset entropy
}
```

```

gain <- entropy_total - weighted_entropy
return(gain)
}

# Calculate info gain for each attribute
cat("The info gain for department is:", info_gain(data, "department"), "\n")

## The info gain for department is: 0.04860679

cat("The info gain for age is:", info_gain(data, "age"), "\n")

## The info gain for age is: 0.4247351

cat("The info gain for salary is:", info_gain(data, "salary"), "\n")

## The info gain for salary is: 0.5375181

```

Problem 4: If-Then Rules from Decision Tree

```

cat(" Root of decision tree is Salary (highest info gain)\n",
    "First split is on Department\n",
    "Age split is not needed\n")

## Root of decision tree is Salary (highest info gain)
## First split is on Department
## Age split is not needed

cat(" If salary = 26k_30k then status = junior\n",
    "If salary = 31k_35k then status = junior\n",
    "If salary = 36k_40k then status = senior\n",
    "If salary = 41k_45k then status = junior\n",
    "If salary = 46k_50k and department = sales then status = senior\n",
    "If salary = 46k_50k and department = systems then status = junior\n",
    "If salary = 46k_50k and department = marketing then status = senior\n",
    "If salary = 66k_70k then status = senior")

## If salary = 26k_30k then status = junior
## If salary = 31k_35k then status = junior
## If salary = 36k_40k then status = senior
## If salary = 41k_45k then status = junior
## If salary = 46k_50k and department = sales then status = senior
## If salary = 46k_50k and department = systems then status = junior
## If salary = 46k_50k and department = marketing then status = senior
## If salary = 66k_70k then status = senior

```