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
```

(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

```r
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]

```r
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", minmax_normalize(age, 10, 20),
```

```
## The min max normalization function with values 10, 20 produces:
##  10 10.35088 10.52632 10.52632 11.05263 11.22807 11.22807 11.40351 11.57895 11.57895 12.10526 12.105
```

```
cat("\n")
```

```
cat("The min max normalization function with values 0.0 , 1.0 produces:\n", minmax_normalize(age, 0, 1)
```

```
## The min max normalization function with values 0.0 , 1.0 produces:
##  0 0.03508772 0.05263158 0.05263158 0.1052632 0.122807 0.122807 0.1403509 0.1578947 0.1578947 0.2105
```

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.

```r
data <- data.frame(
  department = c("sales", "sales", "sales", "systems", "systems", "systems", "systems", "marketing", "ma
  age = c("31_35", "26_30", "31_35", "21_25", "31_35", "26_30", "41_45", "36_40", "31_35", "46_50", "26_
  salary = c("46K_50K", "26K_30K", "31K_35K", "46K_50K", "66K_70K", "46K_50K", "66K_70K", "46K_50K", "4
  status = c("senior", "junior", "junior", "junior", "senior", "junior", "senior", "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
```

```r
# 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
  }

  gain <- entropy_total - weighted_entropy # info gain = total entropy - weighted subset 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
```

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

```
## The info gain for age is: 0.4247351
```

```r
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

```r
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
```

```r
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
```