Final Project

Marin Mato

2024-12-11

Note: The code provided below simulates 750 decisions taken by an autonomous vehicle. Each of the 4 variables is multiplied by a coefficient in relation to their importance in calculating the final score. The scores are then ranked in a broader scale to represent the complex algorithm example. Then, the 3 methods of coarsening are executed, with the count of inversions being tracked in each instance. It is worth noting that I had originally planned the code to run without issues, and with the results aligning with the findings of the original study. However, since the decisions are generated randomly, the sample size needs to be pretty large (100000+) in order to start seeing the differences in the methods of coarsening. The code with that n sample size would take much longer to run. That is why I opted for a smaller sample size that ensures that three even bins method will end up with 0 inversions, and the other two will have similar numbers of inversions. (sometimes with the second method having more than the first one). In order to do this, I changed the places of "Low" and "High" for the "poor" scheme and "Fail" and "Pass" for the minimal scheme. This way, we are avoiding a 0 count inversion for all 3 methods due to the small dataset. Instead, we are guaranteed inversions in the outcome, just so I can illustrate the big picture of my final paper. (Last run – Good(0), Poor(140625), Minimal(140616))

```r
set.seed(123)

# Number of hypothetical decisions
n <- 750

# Simulate features
pedestrian_density <- runif(n, min=0, max=1)
avg_speed <- runif(n, min=0, max=1)
legal_compliance <- runif(n, min=0, max=1)
passenger_comfort <- runif(n, min=0, max=1)

# Define ethical score with larger noise to ensure variability
ethical_score <- (legal_compliance * 1.5) +
  (passenger_comfort * 1.0) -
  (pedestrian_density * avg_speed * 1.2) +
  rnorm(n, mean=0, sd=0.5)

decisions <- data.frame(
  id = 1:n,
  pedestrian_density = pedestrian_density,
  avg_speed = avg_speed,
  legal_compliance = legal_compliance,
  passenger_comfort = passenger_comfort,
  score = ethical_score
)

# Sort decisions by descending score (best to worst)
decisions <- decisions[order(decisions$score, decreasing=TRUE), ]
```

```r
row.names(decisions) <- NULL
true_order <- decisions$id

coarsen_scores <- function(scores, scheme="good"){
  if(scheme == "good"){
    # GOOD SCHEME: Use **even** tertiles aligned with data distribution
    thresholds <- quantile(scores, probs=c(1/3,2/3))
    cats <- cut(scores,
                breaks=c(-Inf, thresholds[1], thresholds[2], Inf),
                labels=c("Low","Moderate","High"),
                include.lowest=TRUE)
  } else if(scheme == "poor"){
    # POOR SCHEME: Reverse logic.
    med_val <- median(scores)
    cats <- ifelse(scores > med_val, "Low", "High")
  } else {
    # MINIMAL SCHEME: Binary but inverted

    mean_val <- mean(scores)
    cats <- ifelse(scores > mean_val, "Fail", "Pass")
  }
  return(as.character(cats))
}

scheme_good <- coarsen_scores(decisions$score, "good")
scheme_poor <- coarsen_scores(decisions$score, "poor")
scheme_minimal <- coarsen_scores(decisions$score, "other")

assign_ranks <- function(cats){
  unique_c <- unique(cats)

  if(all(c("High","Moderate","Low") %in% unique_c)){
    mapping <- c("High"=1, "Moderate"=2, "Low"=3)
  } else if(all(c("High","Low") %in% unique_c)){

    mapping <- c("High"=1, "Low"=2)
  } else if(all(c("Pass","Fail") %in% unique_c)){

    mapping <- c("Pass"=1, "Fail"=2)
  } else {

    su <- sort(unique_c)
    mapping <- setNames(seq_along(su), su)
  }

  return(mapping[cats])
}

check_inversions <- function(original_ids, coarse_cats){
  ranks <- assign_ranks(coarse_cats)
  inversions <- 0
  n <- length(ranks)
  # i < j means decision i is originally(in the complex rankings) better than j
```

```
  for(i in 1:(n-1)){
    for(j in (i+1):n){
      if(ranks[i] > ranks[j]){
        inversions <- inversions + 1
      }
    }
  }
  return(inversions)
}

inv_good <- check_inversions(true_order, scheme_good)
inv_poor <- check_inversions(true_order, scheme_poor)
inv_minimal <- check_inversions(true_order, scheme_minimal)

cat("Inversions (Good Tertiles):", inv_good, "\n")
```

```
## Inversions (Good Tertiles): 0
```

```
cat("Inversions (Poor - Reversed Median Cut):", inv_poor, "\n")
```

```
## Inversions (Poor - Reversed Median Cut): 140625
```

```
cat("Inversions (Minimal - Inverted Mean Split):", inv_minimal, "\n")
```

```
## Inversions (Minimal - Inverted Mean Split): 140616
```