Data_splitting

Hun

2022-03-23

Importing data

Algorithm for splitting variables into training and testing set evenly

```
# My goal is to use quantiles to split data evenly
split <- function(variable, p) {

  #Case for categorical variable or small number of levels of continuosu variable
  if (length(unique(variable)) < 10){
    unique <- unique(variable)
    unique_list <- list()

    training_store <- vector()

    testing_store <- vector()


    for (i in 1:length(unique)) {
      unique_list[[i]] <- variable[variable %in% unique[i]]

      #To make sure all variables of training/testing have the same number
      ifelse((i %% 2) == 1 && (length(variable)*0.8)%%1 > 0.5,
             index <- sample(1:length(unique_list[[i]]),
                      ceiling(length(unique_list[[i]])*0.8),replace=FALSE),
             index <- sample(1:length(unique_list[[i]]),
                      floor(length(unique_list[[i]])*0.8),replace=FALSE))
      #To make sure all variables of training/testing have the same number
      ifelse((i %% 2) == 1 && (length(variable)*0.8)%%1 < 0.5,
             index <- sample(1:length(unique_list[[i]]),
                      floor(length(unique_list[[i]])*0.8),replace=FALSE),
             index <- sample(1:length(unique_list[[i]]),
                      ceiling(length(unique_list[[i]])*0.8),replace=FALSE))


      training_store <- c(training_store, unique_list[[i]][index])
      testing_store <- c(testing_store, unique_list[[i]][-index])
    }
    split <- list(training = training_store, testing = testing_store)
  }
  else {

  #Case for continuous variable
  variable <- sort(variable)

  #Using quantiles to split data evenly
  smallest <- min(variable)
  first_quantile <- variable[round(length(variable)*0.25, 0)]
  second_quantile <- variable[round(length(variable)*0.5, 0)]
  third_quantile <- variable[round(length(variable)*0.75, 0)]
  largest <- max(variable)

  summary <- c(smallest, first_quantile, second_quantile, third_quantile, largest)

  training_data <- list()
  testing_data <- list()

  training_store <- vector()

  testing_store <- vector()

  for(i in 1:(length(summary)-1)){
    #To make sure all variables of training/testing have the same nummber
    ifelse(i == (length(summary)-1),
             Q_data <- variable[variable>=summary[i] & variable<=summary[i+1]],
             Q_data <- variable[variable>=summary[i] & variable<summary[i+1]])
    #To make sure all variables of training/testing have the same nummber
    ifelse((i %% 2) == 1,
             index <- sample(1:length(Q_data), ceiling(length(Q_data)*0.8), replace=FALSE),
             index <- sample(1:length(Q_data), floor(length(Q_data)*0.8), replace=FALSE))


    training_data[[i]] <- Q_data[index]

    testing_data[[i]] <- Q_data[-index]

    training_store <- c(training_store, training_data[[i]])

    testing_store <- c(testing_store, testing_data[[i]])
  }
  split <- list(training = training_store, testing = testing_store)
  }
 return(split)
}
```

```r
#Let's check if the algorithm works for continuous variable

split1 <- split(data$radius_mean, 0.8)

data_frame(mean = mean(split1$training), sd = sd(split1$training),
           proportion = length(split1$training)/nrow(data))
```

```
## # A tibble: 1 x 3
##    mean    sd proportion
##   <dbl> <dbl>      <dbl>
## 1  14.1  3.47      0.800
```

```r
data_frame(mean = mean(split1$testing), sd = sd(split1$testing),
           proportion = length(split1$testing)/nrow(data))
```

```
## # A tibble: 1 x 3
##    mean    sd proportion
##   <dbl> <dbl>      <dbl>
## 1  14.1  3.77      0.200
```

```r
#Looks good
```

```r
#Let's check if the algorithm works for categorical variable
split2 <- split(data$diagnosis, 0.8)

data_frame(mean = mean(split2$training), sd = sd(split2$training),
           proportion = length(split2$training)/length(data$radius_mean))
```

```
## # A tibble: 1 x 3
##    mean    sd proportion
##   <dbl> <dbl>      <dbl>
## 1 0.371 0.484      0.800
```

```r
data_frame(mean = mean(split2$testing), sd = sd(split2$testing),
           proportion = length(split2$testing)/length(data$radius_mean))
```

```
## # A tibble: 1 x 3
##    mean    sd proportion
##   <dbl> <dbl>      <dbl>
## 1 0.377 0.487      0.200
```

```r
#Looks good
```

Applying algorithm to get the training/testing data frame of entire data

```r
data_split <- function(data,split){
  data_split <- map(data, split, 0.8)
  training_list <- list()
  testing_list <- list()
  for (i in 1:length(data_split)) {

    training_list[[i]] <- data_split[[i]]$training
    testing_list[[i]] <- data_split[[i]]$testing
  }

  names(training_list) <- names(data_split)
  names(testing_list) <- names(data_split)

  training <- dplyr::bind_rows(training_list) %>% data.frame()
  testing <- dplyr::bind_rows(testing_list) %>% data.frame()

  return(list(training, testing))
}
```

Combining result together to make it reader-frindly

```
trainging_result <-
  skimr::skim_without_charts(data_split(data,split)[1]) %>% data.frame() %>%
   select(2,5,6) %>%
  rename(training_mean = numeric.mean, training_sd = numeric.sd) %>%
  mutate_if(is.numeric, ~round(.x, digits = 3)) %>%
  mutate_if(is.numeric, ~format(.x, scientific = FALSE))


testing_result <-
  skimr::skim_without_charts(data_split(data,split)[2]) %>%
  select(2,5,6) %>%
  rename(testing_mean = numeric.mean, testing_sd = numeric.sd) %>%
  mutate_if(is.numeric, ~round(.x, digits = 3)) %>%
  mutate_if(is.numeric, ~format(.x, scientific = FALSE))


trainging_result %>% left_join(testing_result, by = "skim_variable") %>% kable()
```

skim_variable	training_mean	training_sd	testing_mean	testing_sd
id	27147102.360	112261857.440	25501012.675	119161044.375
diagnosis	0.371	0.484	0.377	0.487
radius_mean	14.163	3.614	13.995	3.677
texture_mean	19.279	4.369	19.167	3.974
smoothness_mean	0.096	0.015	0.097	0.016
compactness_mean	0.105	0.053	0.104	0.050
concave_points_mean	0.049	0.039	0.049	0.038
symmetry_mean	0.181	0.026	0.181	0.026
fractal_dimension_mean	0.063	0.007	0.063	0.008
radius_se	0.408	0.289	0.412	0.269
texture_se	1.215	0.552	1.240	0.624
smoothness_se	0.007	0.003	0.007	0.002
compactness_se	0.026	0.018	0.026	0.020
concavity_se	0.032	0.032	0.033	0.041
concave_points_se	0.012	0.006	0.012	0.006
symmetry_se	0.021	0.008	0.021	0.009
fractal_dimension_se	0.004	0.003	0.004	0.003
radius_worst	16.240	4.802	16.029	4.566
smoothness_worst	0.132	0.023	0.132	0.023
compactness_worst	0.256	0.162	0.249	0.144
concavity_worst	0.276	0.213	0.277	0.224
symmetry_worst	0.289	0.062	0.290	0.061
fractal_dimension_worst	0.084	0.017	0.083	0.016

training_nrow	testing_nrow	eighty_percent_data_nrow	twenty_percent_data_nrow
455	114	455.2	113.8