1. Suppose that we have age data including the following numbers in sorted order. Then answer the questions below. (5 points each)

age: 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) Use smoothing by bin means to smooth the above data, using a bin depth of 3. Illustrate your steps. Comment on the effect of this technique for the given data.

Step1: Sorted data(The data is in ordered, No need to do anything)

Step2: Divide data into Bin depth of 3

Bin 1: 13, 15, 16

Bin 2: 16, 19, 20

Bin 3: 20, 21, 22

Bin 4: 22, 25, 25

Bin 5: 25, 25, 30

Bin 6: 33, 33, 35

Bin 7: 35, 35, 35

Bin 8: 36, 40, 45

Bin 9: 46, 52, 70

Step3: Calculate the mean for each bin

Bin 1 mean: (13 + 15 + 16) / 3 = 14.67

Bin 2 mean: (16 + 19 + 20) / 3 = 18.33

Bin 3 mean: (20 + 21 + 22) / 3 = 21

Bin 4 mean: (22 + 25 + 25) / 3 = 24

Bin 5 mean: (25 + 25 + 30) / 3 = 26.67

Bin 6 mean: (33 + 33 + 35) / 3 = 33.67

Bin 7 mean: (35 + 35 + 35) / 3 = 35

Bin 8 mean: (36 + 40 + 45) / 3 = 40.33

Bin 9 mean: (46 + 52 + 70) / 3 = 56

Bin 10 mean: 70 (only one value)

Step 3: Replace the values in each bin with the calculated mean

Step 4: Smoothing data by mean:

14.67, 14.67, 14.67, 18.33, 18.33, 18.33, 21, 21, 21, 24, 24, 24, 26.67, 26.67, 26.67, 33.67, 33.67, 33.67, 35, 35, 35, 40.33, 40.33, 40.33, 56, 56, 56

```r
```{r 1#}
#1.(a)
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)
bin_means_smoothing <- function(data, bin_depth) {
  smoothed_data <- numeric(length(data))
  for (i in 1:length(data)) {
    bin_index <- ceiling(i / bin_depth)
    bin_start <- (bin_index - 1) * bin_depth + 1
    bin_end <- min(bin_index * bin_depth, length(data))
    if(bin_end>length(data)){
      bin_end = length(data)
    }
    bin_mean <- mean(data[bin_start:bin_end])
    smoothed_data[bin_start:bin_end] <- bin_mean
  }
  return(smoothed_data)
}
smoothed_age <- bin_means_smoothing(age, 3)
print(smoothed_age)
```
```

```
 [1] 14.66667 14.66667 14.66667 18.33333 18.33333 18.33333 21.00000 21.00000
 [9] 21.00000 24.00000 24.00000 24.00000 26.66667 26.66667 26.66667 33.66667
[17] 33.66667 33.66667 35.00000 35.00000 35.00000 40.33333 40.33333 40.33333
[25] 56.00000 56.00000 56.00000
```

(b) Use IQR measure to determine if there are any outliers in this data.

Inter-quartile range: IQR = Q3 – Q1

Quartiles: Q1 (25th percentile), Q3 (75th percentile)

Outlier: usually, a value higher/lower than 1.5 x IQR

Q1=20.5，Q3=35

IQR = Q3 – Q1 = 35 - 20.5 = 14.5

1.5 x IQR = 14.5 * 1.5 = 21.75

Lower = 20 - 21.75 = - 1.75

Upper = 35 + 21.75 = 56.75

70 is the outlier in the data.

```
```{r 1#}
#1.(b)
summary(age)
fun.outlier <- function(x,time.iqr=1.5) {
  print("IQR:")
  print(IQR(x))
  outlier.low <- quantile(x,probs=c(0.25))-IQR(x)*time.iqr
  outlier.high <- quantile(x,probs=c(0.75))+IQR(x)*time.iqr
  print("outliers:")
  x[which(x>outlier.high | x<outlier.low)]
}
print(fun.outlier(age))
```
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  13.00   20.50   25.00   29.96   35.00   70.00
[1] "IQR:"
[1] 14.5
[1] "outliers:"
[1] 70
```

(c) Use min-max normalization to transform the value 35 for age onto the range [0.0, 1.0].

Normalization(min-max)=((35-min)/(max-min))*(new_max-new_min)+new_min

= ((35-13)/(70-13)) *(1.0-0.0)+0.0= 0.386

```
```{r 1#}
#1.(c)
# min-max
min_max <- ((35 - min(age)) / (max(age) - min(age))) * (1.0 - 0.0) + 0.0
print(min_max)
```
```

```
[1] 0.3859649
```

(d) Use z-score normalization to transform the value 35 for age? (you need to compute mean and standard deviation first)

mean = (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)/27=29.96296

sd = 12.94212

Normalization(z-score) = (35-mean)/sd = (35-29.96296)/ 12.94212 = 0.3891971

```r
#1.(d)
#
mean_age <- mean(age)
sd_age <- sd(age)
z_score <- (35 - mean_age) / sd_age
print(mean_age)
print(sd_age)
print(z_score)
```

```
[1] 29.96296
[1] 12.94212
[1] 0.3891971
```

(e) Use normalization by decimal scaling to transform the value 35 for age.

Max value is 70, 10^1 is less than 70, 10^2 is over 70, so we choose 10^2.

Normalization(decimal scaling) = 35/10^2 = 35/ 100 = 0.35

```r
#1.(e)
age_value <- 35
# Find the appropriate power of 10
power <- ceiling(log10(max(age)))
# Perform decimal scaling normalization
normalized_age <- age_value / (10 ^ power)
# Print the normalized value
print(normalized_age)
```

```
[1] 0.35
```

2. Write a function in your preferred language which can take a data vector and do min-max normalization by transforming data onto a desired range. For example, it should be able get the age data above and map it between any two numbers. (foo (a, min_new, max_new) where a is an one-dimensional array) (25 points)

```r
#2.
foo <- function(data, min_new, max_new) {
  min <- min(data)
  max <- max(data)
  normalized_min_max_data <- ((data - min) / (max - min)) * (max_new - min_new) + min_new
  return(normalized_min_max_data)
}

normalized_min_max_age <- foo(age, 0, 1)
print(normalized_min_max_age)
```

```
 [1] 0.00000000 0.03508772 0.05263158 0.05263158 0.10526316 0.12280702 0.12280702 0.14035088 0.15789474
[10] 0.15789474 0.21052632 0.21052632 0.21052632 0.21052632 0.29824561 0.35087719 0.35087719 0.38596491
[19] 0.38596491 0.38596491 0.38596491 0.40350877 0.47368421 0.56140351 0.57894737 0.68421053 1.00000000
```

department	age	salary	status	count
sales	31_35	46K_50K	senior	30
sales	26_30	26K_30K	junior	40
sales	31_35	31K_35K	junior	40
systems	21_25	46K_50K	junior	20
systems	31_35	66K_70K	senior	5
systems	26_30	46K_50K	junior	3
systems	41_45	66K_70K	senior	3
marketing	36_40	46K_50K	senior	10
marketing	31_35	41K_45K	junior	4
secretary	46_50	36K_40K	senior	4
secretary	26_30	26K_30K	junior	6

Table 1: Data shows the count of each feature combination. For instance, There are 30 senior sales staff who are 31...35 years old and have 46...50K salary. Since each combination is unique, their corresponding groups are mutually exclusive which implies counts are not double counts for any of the cases. Notice that the status column is the class label to indicate whether someone is junior or senior.

3. Using information gain on the data in Table 1, do calculations for two levels of a decision tree which decides whether a person is senior or junior. Please show your calculations and clearly write down your junior and senior counts not to confuse yourself. Note that you need to calculate the information gain for all attributes (department, age, salary) and pick the one to start your tree. In your subsets of your data, you'll perform the same operation for the attributes available. (You can use a computing environment to write the mathematical expressions. i.e., p* logp,(1/2)* log2(1/2)) (25pts)

Info(D) = -(log2(113/165) * 113/165 + log2(52/165) * 52/165) = 0.899

D	age	junior	senior	Total
D1	21_25	20	0	20
D2	26_30	49	0	49
D3	31_35	44	35	79
D4	36_40	0	10	10
D5	41_45	0	3	3
D6	46_50	0	4	4
	Total	113	52	165

Info-age(D)=-(log2(20/20)*20/20+log2(0)*0)*20/165-(log2(49/49)*49/49+log2(0)*0)*49/165-(log2(44/79)*44/79+log2(35/79)*35/79)*79/165-(log2(0)*0+log2(10/10)*10/10)*10/165-(log2(0)*0+log2(3/3)*3/3)*3/165-(log2(0)*0+log2(4/4)*4/4)*4/165=0.474

Gain(age) = Info(D) - Info-age(D) = 0.899 - 0.474 = 0.425

D	department	junior	senior	Total
D1	sales	80	30	110
D2	systems	23	8	31
D3	marketing	4	10	14
D4	secretary	6	4	10
	Total	113	52	165

Info-department(D)=-(log2(80/110)*80/110+log2(30/110)*30/110)*110/165-(log2(23/31)*23/31+log2(8/31)*8/31)*31/165-(log2(4/14)*4/14+log2(10/14)*10/14)*14/165-(log2(6/10)*6/10+log2(4/10)*4/10)*10/165=0.85

Gain(department) = Info(D) - Info-age(D) = 0.899 - 0.85 = 0.049

D	salary	junior	senior	count
D1	26K_30K	46	0	46
D2	31K_35K	40	0	40
D3	36K_40K	0	4	4
D4	41K_45K	4	0	4
D5	46K_50K	23	40	63
D6	66K_70K	0	8	8
	Total	113	52	165

Info-salary(D)=-(log2(46/46)*46/46+log2(0)*0)*46/165-(log2(40/40)*40/40+log2(0)*0)*40/165-(log2(0)*0+log2(4/4)*4/4)*4/165-(log2(4/4)*4/4+log2(0)*0)*4/165-(log2(23/63)*23/63+log2(40/63)*40/63)*63/165-(log2(0)*0+log2(8/8)*8/8)*8/165=0.362

Gain(salary) = Info(D) - Info-age(D) = 0.899 - 0.362 = 0.537

Gain(salary) is the highest one, so salary is the root of this tree then age and department.

4. Using the decision tree you generate if-then rules. (25pts)

IF salary = 26K_35K THEN status = junior

IF salary = 36K_40K AND age = 46_50 THEN status = senior

IF salary = 41K_45K AND age = 31_35 THEN status = junior

IF salary = 46K_50K AND age = 21_30 THEN status = junior

IF salary = 46K_50K AND age = 31_40 THEN status = senior

IF salary = 66K_70K THEN status = senior