1. Find the distance between objects 1 and 3 by using the formula provided on the slides. Notice that we have mixed type of attributes.

Object Identifier	test-1(nominal)	test-2 (ordinal)	test-3 (numeric)
1	A	excellent	45
2	B	fair	22
3	C	good	64
4	A	excellent	28

1 (A, excellent, 45)

3 (C, good, 64)

Nominal:

m: # of matches, p: total # of variables

m=0,p=1

$$d_{13} = \frac{p-m}{p} = \frac{1-0}{1} = 1$$

$$\delta_{13}^{(f)} = 1$$

Ordinal:

fair = 1, good = 2, excellent = 3

fair = 1-1/3-1 = 0, good = 2-1/3-1 = 0.5, excellent = 3-1/3-1 = 1

$$d_{13} = |1-0.5| = 0.5$$

$$\delta_{13}^{(f)} = 1$$

Numeric:

$$d_{13} = \frac{|45-64|}{64-22} = \frac{19}{42}$$

$$\delta_{13}^{(f)} = 1$$

Distance between 1 and 3:

$$d(1,3) = \frac{1+0.5+\frac{19}{42}}{1+1+1} = \frac{\frac{82}{42}}{3} = 0.65$$

2. Write a program in any language which can compute Manhattan and Euclidean distances between any two given vectors with any length. You can pass the length to your function, but please don't limit the dimension to 2. You can test your function on vectors you fill in your code without asking user input.

```
#Create function for Manhattan and Euclidean distances.
Manhattan_Distance <- function(vector1, vector2) {
   sum(abs(vector1 - vector2))
}
Euclidean_Distance <- function(vector1, vector2) {
   sqrt(sum((vector1 - vector2)^2))
}

#Create 2 vectors.
vector_s1 <- c(9, 7, 3, 10, 25)
vector_s2 <- c(4, 10, 30, 22, 100)
#Get the result use the distance function with the vectors
M_Result <- Manhattan_Distance(vector_s1, vector_s2)
E_Result <- Euclidean_Distance(vector_s1, vector_s2)

#Print the result
cat("Manhattan Distances: ", M_Result, "\n")
cat("Euclidean Distances: ", E_Result, "\n")
```

Manhattan Distance = |9-4|+|7-10|+|3-30|+|10-22|+|2-100|=122

Euclidean Distance = $\sqrt{(9-4)^2+(7-10)^2+(3-30)^2+(10-22)^2+(2-100)^2}$

=80.82079

```
> #Create function for Manhattan and Euclidean distances
> Manhattan_Distance <- function(vector1, vector2) {
+   sum(abs(vector1 - vector2))
+ }
>
> Euclidean_Distance <- function(vector1, vector2) {
+   sqrt(sum((vector1 - vector2)^2))
+ }
> #Create 2 vectors
> vector_s1 <- c(9, 7, 3, 10, 25)
> vector_s2 <- c(4, 10, 30, 22, 100)
> #Get the result use the distance function with the vectors
> M_Result <- Manhattan_Distance(vector_s1, vector_s2)
> E_Result <- Euclidean_Distance(vector_s1, vector_s2)
> #Print the result
> cat("Manhattan Distances: ", M_Result, "\n")
Manhattan Distances:  122
> cat("Euclidean Distances: ", E_Result, "\n")
Euclidean Distances:  80.82079
```

3. In the table below, determine whether passing a class has a dependency on attendance by using Chi-square test. Please refer to the formula in the slides.

	Passed		Failed		**Total**
	Real	Expected	Real	Expected	
Attended	25.00	18.94	6.00	12.06	**31.00**
Skipped	8.00	14.06	15.00	8.94	**23.00**
Total	**33.00**		**21.00**		**54.00**

$$\chi^2 = \frac{(25-18.94)^2}{18.94} + \frac{(8-14.06)^2}{14.06} + \frac{(6-12.06)^2}{12.06} + \frac{(15-8.94)^2}{8.94} = 11.70$$

degree of freedom (2-1) * (2-1) =1

X2 (1, N = 54) = 11.7, p < .01

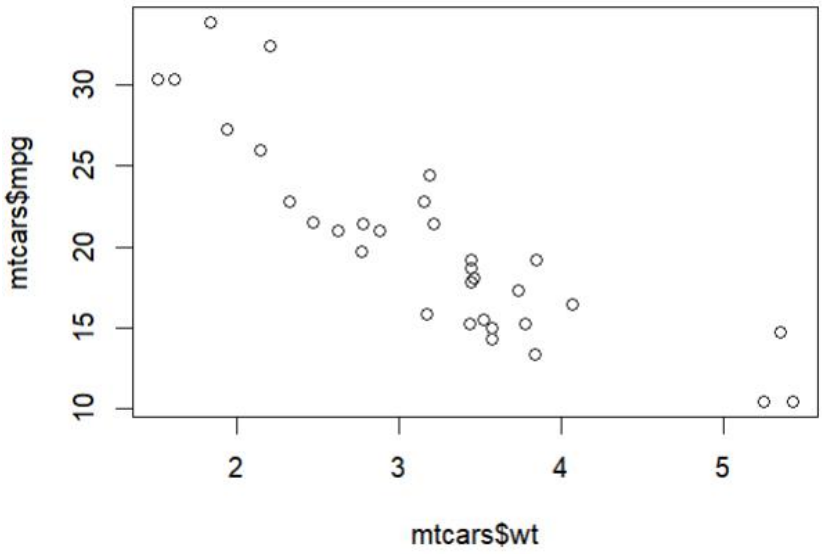
So passing a class has a dependency on attendance.

4. In R, there is a built-in data frame called mtcars. Please calculate the correlation between mpg and wt attributes of mtcars by using cor() function. Then generate scatter plot based on these two attributes. Your scatter plot should be like the one below. You don't need to submit the image, but R script should be submitted

cor(mtcars$mpg,mtcars$wt)

```
> cor(mtcars$mpg,mtcars$wt)
[1] -0.8676594
```

scatter.smooth(x=mtcars$wt,y=mtcars$mpg,evaluation = 0)



5. Grad Students Only Write an R or Python script which removes or drops the columns which have more than 75% missing values. Then it should replace the missing values in the remaining columns with the median value of the existing values of that particular column. Download metabolite.csv from Google Drive and use this data set to test your code. Please check the end of this document for some useful R examples and hints. (10 points)

Total 192 column.

There are 4 columns which have more than 75% missing values.

Drop column Names:

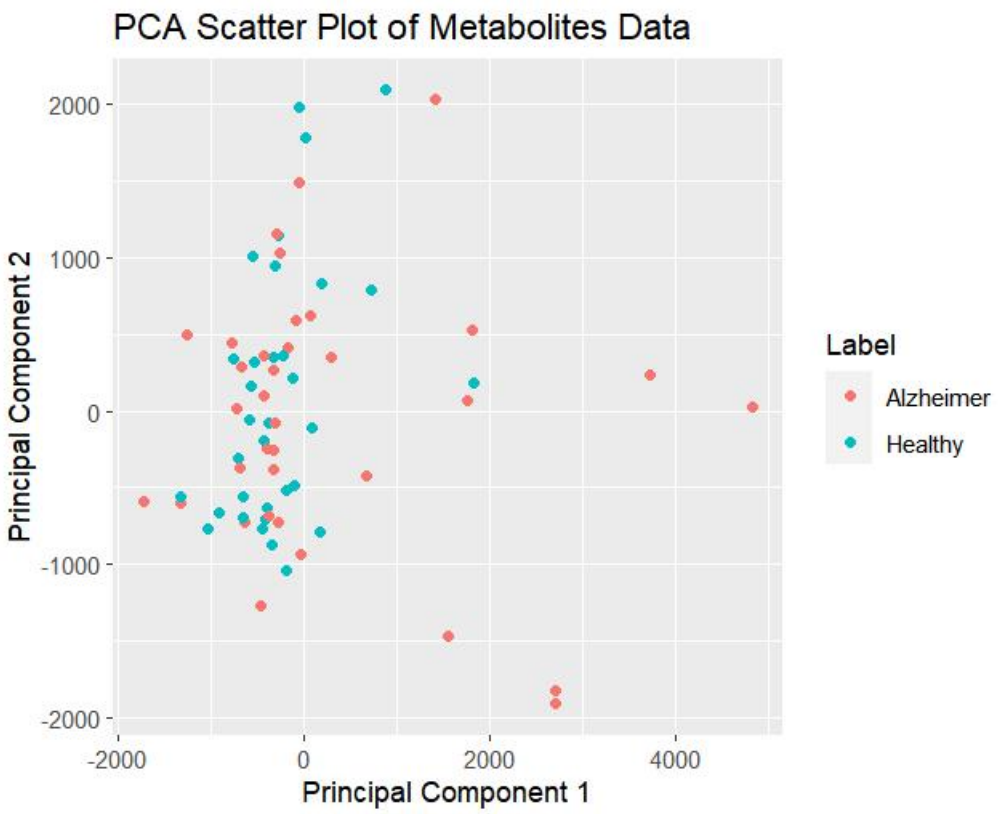
```
[1] "Nitro.Tyr"
[1] "PEA"
[1] "Spermine"
[1] "PC.ae.C38.1"
```

```
#Start get library and read the files
library(tidyverse)
setwd("C:\\Users\\DELL\\Desktop\\CSC587\\datamining-main\\Rscripts")
data.metabolite <- file.path('data', 'metabolite.csv')
metabolite = read.delim(data.metabolite, sep=',', header = T)

#Get the list of the columns which will not be removed.
columns_remain <- NULL
for (i in 1:ncol(metabolite)) {
  missing_proportion <- sum(is.na(metabolite[,i])) / length(metabolite[,i])
  if(missing_proportion>0.75){
    drop_column_name <- colnames(metabolite)[i]
    print(drop_column_name)
  }else{
    remain_column_name <- colnames(metabolite)[i]
```

```
        columns_remain <- c(columns_remain,remain_column_name)
    }
}
metabolite <- subset(metabolite,select=columns_remain)

#Fill the missing values with the median value in their own columns
for (i in 1:ncol(metabolite)) {
    median_value <- median(metabolite[,i],na.rm = TRUE)
    for (j in 1:nrow(metabolite)) {
        if(is.na(metabolite[j,i])){
            metabolite[j,i] <- median_value
        }
    }
}
print(metabolite)
```

6. Grad Students Only Please apply PCA on the processed metabolites data and create a scatter plot by using first two principal components in which points are colored based on the Label column. Please submit your code along with your figure in the same file.

```
# Subset Label column
metabolite_pca <- subset(metabolite,select=-Label)
pca_apply <- prcomp(metabolite_pca)
print(pca_apply)
pc_scores <- as.data.frame(pca_apply$x[, 1:2])
# Add back the Label column
pc_scores$Label <- metabolite$Label
print(pc_scores)
ggplot(pc_scores, aes(x = PC1, y = PC2, color = Label)) +
    geom_point() +
    labs(title = "PCA Scatter Plot of Metabolites Data",
             x = "Principal Component 1",
             y = "Principal Component 2")
```


PCA Scatter Plot of Metabolites Data
Principal Component 2
2000
1000
0
-1000
-2000
-2000
0
2000
4000
Principal Component 1
Label
Alzheimer
Healthy

	PC1	PC2	Label
1	-179.20084	416.88290	Alzheimer
2	-1323.26135	-604.19473	Alzheimer
3	-647.24841	-730.06345	Alzheimer
4	-1719.11694	-586.89472	Alzheimer
5	-461.84103	-1273.68357	Alzheimer
6	-728.83748	14.82042	Alzheimer
7	-104.46543	-487.11446	Healthy
8	-664.43109	-696.05942	Healthy
9	-401.39663	-630.04775	Healthy
10	188.99259	831.00122	Healthy
11	-309.52815	944.75819	Healthy
12	-55.18537	1981.99127	Healthy
13	-231.73627	366.51405	Healthy
14	-548.78165	1009.50825	Healthy
15	-714.31520	-310.06349	Healthy
16	13.64383	1786.04919	Healthy
17	79.76400	-113.67639	Healthy
18	-922.76155	-666.28719	Healthy
19	-667.23962	288.07833	Alzheimer
20	-1036.38724	-771.23628	Healthy
21	1828.04151	183.84455	Healthy
22	-331.45522	355.26604	Healthy
23	75.35838	623.97927	Alzheimer
24	-584.71658	-60.24195	Healthy
25	-349.92694	-876.69197	Healthy
26	-188.15030	-522.70932	Healthy
27	-450.57621	-770.53553	Healthy
28	-198.28413	-1044.91877	Healthy
29	727.71263	791.79114	Healthy
30	-1325.71445	-559.53500	Healthy
31	-282.85694	1150.90480	Healthy
32	-437.05400	-190.15593	Healthy
33	169.46843	-793.34038	Healthy
34	-416.02597	-707.41408	Healthy
35	-407.03089	-706.82359	Healthy
36	-648.42173	-564.55869	Healthy
37	-539.57701	315.32588	Healthy
38	-400.24569	-249.65816	Alzheimer
39	-95.01486	591.82628	Alzheimer