Data Wrangling (Data Preprocessing)

Practical assessment 1

Patrick Adrianus

Setup

```
# Load the necessary packages required to reproduce the report. For example:

library(kableExtra)
library(magrittr)
```

Student names, numbers and percentage of contributions

Group information

Student name	Student number	Percentage of contribution
Patrick Adrianus	s3967271	50%
Anjali Lata Rambarath	s3943075	50%

Data Description

This dataset is sourced from https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset (https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset)

According to the requirements, we needed one numeric and one categorical data. We decided to chose a dataset pertaining to 12 attributes, with six categorical variables and six numerical variables

It have 12 Attributes which include id, gender, age, hypertension, heart_disease, ever_married, work_type, Residence_type, avg_glucose_level, bmi, smoking_status, stroke

- ID : Unique Identifier (int)
- Gender : “Male”, “Female”, or “Other” (Factor)
- Age : The age of the patient (num)
- Hypertension : 0 if the patient doesn’t have hypertension and 1 if the patient have hypertention (int)
- heart_disease : 0 if the patient doesn’t have heart disease and 1 if the patient have hear disease (int)
- ever_married : “No” or “Yes” (Factor)
- work_type : “children”, “Govt_jov”, “Never_worked”, “Private”, or “Self-employed” (Factor)
- Residence_type : “Rural” or “Urban” (Factor)
- avg_glucose_level : average glucose level in patient’s blood (num)
- bmi : body mass index (num)
- smoking_status : “formerly smoked”, “never smoked”, “smokes”, or “Unknown” (Factor)
- stroke : 1 if the patient had a stroke or 0 if not (int)

Read/Import Data

- Download a stroke dataset in a local file directory.
- The ‘read.csv’ function is used to read the healthcare-dataset-stroke-data.csv file. The ‘stringasFactors’ is set to TRUE to make sure any columns containing text data be converted into factors (categorical values).
- ‘head()’ function is used to display the first few rows of the data to examine the structure and content of the data.

```
# Import the data, provide your R codes here.

data <- read.csv("healthcare-dataset-stroke-data.csv", stringsAsFactors = TRUE)
head(data)
```

	id <int>	gender <fct>	a… <dbl>	hypertension <int>	heart_disease <int>	ever_married <fct>	work_type <fct>	Residence_type <fct>
1	9046	Male	67	0	1	Yes	Private	Urban
2	51676	Female	61	0	0	Yes	Self-employed	Rural
3	31112	Male	80	0	1	Yes	Private	Rural
4	60182	Female	49	0	0	Yes	Private	Urban
5	1665	Female	79	1	0	Yes	Self-employed	Rural
6	56669	Male	81	0	0	Yes	Private	Urban

6 rows | 1-9 of 13 columns

Inspect and Understand

After importing the data we need to inspect the columns to better understand of the data.

```
# Inspection of your data, provide R codes here.
dim(data)
```

```
## [1] 5110   12
```

```
colnames(data)
```

```
##  [1] "id"                "gender"            "age"
##  [4] "hypertension"      "heart_disease"     "ever_married"
##  [7] "work_type"         "Residence_type"    "avg_glucose_level"
## [10] "bmi"               "smoking_status"    "stroke"
```

```
str(data)
```

```
## 'data.frame':    5110 obs. of  12 variables:
##  $ id               : int  9046 51676 31112 60182 1665 56669 53882 10434 27419 60491 ...
##  $ gender           : Factor w/ 3 levels "Female","Male",..: 2 1 2 1 1 2 2 1 1 1 ...
##  $ age              : num  67 61 80 49 79 81 74 69 59 78 ...
##  $ hypertension     : int  0 0 0 0 1 0 1 0 0 0 ...
##  $ heart_disease    : int  1 0 1 0 0 0 1 0 0 0 ...
##  $ ever_married     : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 1 2 2 ...
##  $ work_type        : Factor w/ 5 levels "children","Govt_job",..: 4 5 4 4 5 4 4 4 4 4 ...
##  $ Residence_type   : Factor w/ 2 levels "Rural","Urban": 2 1 1 2 1 2 1 2 1 2 ...
##  $ avg_glucose_level: num  229 202 106 171 174 ...
##  $ bmi              : Factor w/ 419 levels "10.3","11.3",..: 240 419 199 218 114 164 148 102 419 116 ...
##  $ smoking_status   : Factor w/ 4 levels "formerly smoked",..: 1 2 2 3 2 1 2 2 4 4 ...
##  $ stroke           : int  1 1 1 1 1 1 1 1 1 1 ...
```

```
data$hypertension <- as.logical(data$hypertension)

data$heart_disease <- as.logical(data$heart_disease)

data$bmi <- as.numeric(as.character(data$bmi))
data$bmi[is.na(data$bmi)] <- NA

data$stroke <- as.logical(data$stroke)

#showing the levels of the factor class type
levels(data$gender)
```

```
## [1] "Female" "Male"   "Other"
```

```
levels(data$ever_married)
```

```
## [1] "No"  "Yes"
```

```
levels(data$work_type)
```

```
## [1] "children"      "Govt_job"      "Never_worked"  "Private"
## [5] "Self-employed"
```

```
levels(data$Residence_type)
```

```
## [1] "Rural" "Urban"
```

```
levels(data$smoking_status)
```

```
## [1] "formerly smoked" "never smoked"    "smokes"          "Unknown"
```

```
head(data)
```

	id <int>	gender <fct>	a… <dbl>	hypertension <lgl>	heart_disease <lgl>	ever_married <fct>	work_type <fct>	Residence_type <fct>
1	9046	Male	67	FALSE	TRUE	Yes	Private	Urban

	id <int>	gender <fct>	a... <dbl>	hypertension <lgl>	heart_disease <lgl>	ever_married <fct>	work_type <fct>	Residence_type <fct>
2	51676	Female	61	FALSE	FALSE	Yes	Self-employed	Rural
3	31112	Male	80	FALSE	TRUE	Yes	Private	Rural
4	60182	Female	49	FALSE	FALSE	Yes	Private	Urban
5	1665	Female	79	TRUE	FALSE	Yes	Self-employed	Rural
6	56669	Male	81	FALSE	FALSE	Yes	Private	Urban

6 rows | 1-9 of 13 columns

```
str(data)
```

```
## 'data.frame':    5110 obs. of  12 variables:
##  $ id               : int  9046 51676 31112 60182 1665 56669 53882 10434 27419 60491 ...
##  $ gender           : Factor w/ 3 levels "Female","Male",..: 2 1 2 1 1 2 2 1 1 1 ...
##  $ age              : num  67 61 80 49 79 81 74 69 59 78 ...
##  $ hypertension     : logi  FALSE FALSE FALSE FALSE TRUE FALSE ...
##  $ heart_disease    : logi  TRUE FALSE TRUE FALSE FALSE FALSE ...
##  $ ever_married     : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 1 2 2 ...
##  $ work_type        : Factor w/ 5 levels "children","Govt_job",..: 4 5 4 4 5 4 4 4 4 4 ...
##  $ Residence_type   : Factor w/ 2 levels "Rural","Urban": 2 1 1 2 1 2 1 2 1 2 ...
##  $ avg_glucose_level: num  229 202 106 171 174 ...
##  $ bmi              : num  36.6 NA 32.5 34.4 24 29 27.4 22.8 NA 24.2 ...
##  $ smoking_status   : Factor w/ 4 levels "formerly smoked",..: 1 2 2 3 2 1 2 2 4 4 ...
##  $ stroke           : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
```

- Checking the dimension of the imported data using 'dim()' function which output to 5110 rows and 12 columns.
- Checking the column name of the imported data using 'colnames()' function.
- Checking the structure of the imported data using 'str()' function.
- For the 'hypertension' column we use 'as.logical' function to converts the column from an integer to a logical variable. Logical variables have a value of TRUE or FALSE.
- For the 'heart_disease' column we use 'as.logical' function converts the column from an integer to a logical variable as the 'hypertension' column
- For the 'bmi' column we use 'as.character' function first to converts the column to character and then use the 'as.numeric' to converts it to numerical values. After that, 'is.na' function is used to replaced the missing or invalid values to 'NA'.
- For the 'stroke' column we use 'as.logical' function converts the column from an integer to a logical variable as the 'hypertension' and 'heart_disease' column
- Using the levels() function, we can see the different levels according to the column names and identify if it is relevant to our current data, where no further changes are needed. We recognise the data as a number according to the level that is associated with it
- 'head()' function is used to display the first few rows of the data to examine the structure and content of the data after values and variable type manipulation.
- 'str()' function is used to display the structures of the modified data.

Subsetting

```
# Subset your data and convert it to a matrix, provide R codes here.
subset_data <- data[1:10, ]
matrix_data <- as.matrix(subset_data)
dim(matrix_data)
```

```
## [1] 10 12
```

```
str(matrix_data)
```

```
##  chr [1:10, 1:12] " 9046" "51676" "31112" "60182" " 1665" "56669" "53882" ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:10] "1" "2" "3" "4" ...
##   ..$ : chr [1:12] "id" "gender" "age" "hypertension" ...
```

Provide explanations here.

- Subsetting the data by the rows allows us to present ten observations.
- We use the initial data frame, and using the square brackets [], we can subset the data by the rows. We specify the rows to the comma's left and leave the comma's right blank, indicating that we want all the columns/column names.
- As we want to convert the subsetted data into a matrix, we use the as.matrix() function to convert it whilst specifying the variable we are altering.
- To ensure that our dimensions are what we want and are correct, we use the dim() function, which presents (10 12). There are ten rows and twelve columns where no additional changes are needed.
- Followed by the str() function to present the variable names, class type and the variables to make sure they are what we want
- The structure shows that the class type for all the attributes is all chr (character); although there's a mix of class types in the initial data, the as.matrix() function makes all the variable's class types become one to make it easier to produce calculations later on.

Create a new Data Frame

```
# Create a new data frame, provide R codes here.
#Creating two vectors, one ordinal(adding levels and sprcifying order=TRUE)and one numerical
gradeLevel_vector <- factor(c("HD", "C", "DI",  "NN",  "PA", "PA", "C", "DI", "PA", "C"), levels = c("NN", "PA", "C", "DI",
"HD"), order = TRUE)
mark <- c(90, 65, 78, 44, 50, 53, 62, 79, 58, 63)

#allows you to see the class type and convert if needed
class(gradeLevel_vector)
```

```
## [1] "ordered" "factor"
```

```
class(mark)
```

```
## [1] "numeric"
```

```
#changing the mark vector to an integer
mark_vector<- as.integer(mark)
class(mark_vector)
```

```
## [1] "integer"
```

```
#Producing data frame and checking the class type
#As well as ordering the vectors into another DF
gradesDf <- data.frame(gradeLevel_vector, mark_vector)
df <- gradesDf[order(gradesDf$gradeLevel_vector), ]
levels(df$gradeLevel_vector)
```

```
## [1] "NN" "PA" "C"  "DI" "HD"
```

```
str(df)
```

```
## 'data.frame':    10 obs. of  2 variables:
##  $ gradeLevel_vector: Ord.factor w/ 5 levels "NN"<"PA"<"C"<..: 1 2 2 2 3 3 3 4 4 5
##  $ mark_vector      : int  44 50 53 58 65 62 63 78 79 90
```

```
head(df)
```

	gradeLevel_vector <ord>	mark_vector <int>
4	NN	44
5	PA	50
6	PA	53
9	PA	58
2	C	65
7	C	62

6 rows

```
#Making sure that the class type of anotherVector is numeric
previousmark <-c(83, 70, 55, 50, 72, 70, 83, 87, 50, 57)
class(previousmark)
```

```
## [1] "numeric"
```

```
previousmark <- as.numeric(previousmark)
class(previousmark)
```

```
## [1] "numeric"
```

```
#use cbind() to add vector to the data frame
#againg ordering the data frame and showing the first 6 rows of the unordered data frame and ordered data frame
grades <- cbind(gradesDf, previousmark)
ordered_grades <- grades[order(grades$gradeLevel_vector),]
levels(ordered_grades$gradeLevel_vector)
```

```
## [1] "NN" "PA" "C"  "DI" "HD"
```

```
head(grades)
```

	gradeLevel_vector <ord>	mark_vector <int>	previousmark <dbl>
1	HD	90	83
2	C	65	70
3	DI	78	55
4	NN	44	50
5	PA	50	72
6	PA	53	70

6 rows

```
str(ordered_grades)
```

```
## 'data.frame':    10 obs. of  3 variables:
##  $ gradeLevel_vector: Ord.factor w/ 5 levels "NN"<"PA"<"C"<..: 1 2 2 2 3 3 3 4 4 5
##  $ mark_vector      : int  44 50 53 58 65 62 63 78 79 90
##  $ previousmark     : num  50 72 70 50 70 83 57 55 87 83
```

```
head(ordered_grades)
```

	gradeLevel_vector <ord>	mark_vector <int>	previousmark <dbl>
4	NN	44	50
5	PA	50	72
6	PA	53	70
9	PA	58	50
2	C	65	70
7	C	62	83

6 rows

Provide explanations here.

- Two variables are created, one containing ordinal variables, the “gradeLevel_vector”, and the other containing numerical variables, the “mark.”
- According to the specifications, we want the ordinal variables to be in the factor class type and numerical variables to be in the integer class type. We used the class() function to check the ordinal variables class type
- By using the levels() function, we can see the different levels according to the column names and identify if it is relevant to our current data, where no further changes are needed
- We then implemented the factor() function to change the variable to a factor and as.integer() function to change our numerical data to an integer. gradeDf is our new variable to represent our data frame by using the data.frame() function and adding the necessary variables within it.
- We created another variable and implemented the order() function using the initial data frame and the ordinal variable (gradeLevel_vector) to make sure the data frame was ordered correctly
- Again, we use the str() function to display the database’s internal workings, check the class types, and make changes if necessary.
- As well as utilising the head() function to display the first six rows to make sure that the displayed data is correct and shows two columns of variables, where there is one ordinal column and one numerical column
- We then add another variable of numerical data called “previousmark”, checking the type with the class() function and changing it to display only numeric data using the as.numeric() function.
- Using the cbind() function with the initial data frame (gradesDF) and the other variable (previousmark) where it is then saved to a new vector and ordered using the order() function using the “grades” data frame and the “gradeLevel_vector” where it’s then ranked according to its level
- First, we use the head() function on grades to show the unordered data and then use str() to show the internal workings and head() to show the ordered data.