PRACTICAL – 6

Aim: Implementation of Data Preprocessing techniques.

Theory:

Data preprocessing is the process of transforming raw data into a useful, understandable format. Real-world or raw data usually has inconsistent formatting, human errors, and can also be incomplete. Data preprocessing resolves such issues and makes datasets completer and more efficient to perform data analysis.

In other words, data preprocessing is transforming data into a form that computers can easily work on. It makes data analysis or visualization easier and increases the accuracy and speed of the machine learning algorithms that train on the data.

Why is data preprocessing required?

As you know, a database is a collection of data points. Data points are also called observations, data samples, events, and records.

Each sample is described using different characteristics, also known as **features** or **attributes**. Data preprocessing is essential to effectively build models with these features.

Numerous problems can arise while collecting data. You may have to aggregate data from different data sources, leading to mismatching data formats, such as integer and float.

If you’re aggregating data from two or more independent datasets, the gender field may have two different values for men: man, and male. Likewise, if you’re aggregating data from ten different datasets, a field that’s present in eight of them may be missing in the rest two.

By preprocessing data, we make it easier to interpret and use. This process eliminates inconsistencies or duplicates in data, which can otherwise negatively affect a model’s accuracy. Data preprocessing also ensures that there aren’t any incorrect or missing values due to human error or bugs. In short, employing data preprocessing techniques makes the database completer and more accurate.

There are four stages of data processing: cleaning, integration, reduction, and transformation.

1. Data cleaning

Data cleaning or cleansing is the process of cleaning datasets by accounting for missing values, removing outliers, correcting inconsistent data points, and smoothing noisy data. In essence, the motive behind data cleaning is to offer complete and accurate samples for machine learning models.

2. Data integration

Since data is collected from various sources, **data integration** is a crucial part of data preparation. Integration may lead to several inconsistent and redundant data points, ultimately leading to models with inferior accuracy.

3. Data reduction

As the name suggests, **data reduction** is used to reduce the amount of data and thereby reduce the costs associated with data mining or data analysis. It offers a condensed representation of the dataset. Although this step reduces the volume, it maintains the integrity of the original data. This data preprocessing step is especially crucial when working with big data as the amount of data involved would be gigantic.

4. Data transformation

Data transformation is the process of converting data from one format to another. In essence, it involves methods for transforming data into appropriate formats that the computer can learn efficiently from.

In this practical we are going to implement data preprocessing techniques like:

- Naming and Renaming variables, adding a new variable.
- Dealing with missing data.
- Dealing with categorical data.
- Data reduction using sub setting

Setting working directory

```
> getwd()
[1] "C:/Users/Suraj/Documents"
> 
```

```
> my_data<-mtcars
> head(my_data,5)
                   mpg cyl disp  hp drat    wt  qsec vs am
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0
                  gear carb
Mazda RX4            4    4
Mazda RX4 Wag        4    4
Datsun 710           4    1
Hornet 4 Drive       3    1
Hornet Sportabout    3    2
> my_data1 <- my_data[1:6,1:5]
> my_data1
                   mpg cyl disp  hp drat
Mazda RX4         21.0   6  160 110 3.90
Mazda RX4 Wag     21.0   6  160 110 3.90
Datsun 710        22.8   4  108  93 3.85
Hornet 4 Drive    21.4   6  258 110 3.08
Hornet Sportabout 18.7   8  360 175 3.15
Valiant           18.1   6  225 105 2.76
```

Renaming variable.

```
> getwd()
[1] "C:/Users/Suraj/Documents"
> |
```

```
> my_data1 = rename(my_data1, horse_power = hp)
> my_data1
                   mpg cyl disp horse_power drat
Mazda RX4         21.0   6  160         110 3.90
Mazda RX4 Wag     21.0   6  160         110 3.90
Datsun 710        22.8   4  108          93 3.85
Hornet 4 Drive    21.4   6  258         110 3.08
Hornet Sportabout 18.7   8  360         175 3.15
Valiant           18.1   6  225         105 2.76
> my_data1$new_hp1 <- my_data1$horse_power * 0.5
> colnames(my_data1)
[1] "mpg"         "cyl"         "disp"        "horse_power"
[5] "drat"        "new_hp1"
> my_data1
                   mpg cyl disp horse_power drat new_hp1
Mazda RX4         21.0   6  160         110 3.90    55.0
Mazda RX4 Wag     21.0   6  160         110 3.90    55.0
Datsun 710        22.8   4  108          93 3.85    46.5
Hornet 4 Drive    21.4   6  258         110 3.08    55.0
Hornet Sportabout 18.7   8  360         175 3.15    87.5
Valiant           18.1   6  225         105 2.76    52.5
```

Naming variable

```
  formal argument "file" matched by multiple actual arguments
> data2 = read.table(file ="C:/Users/Suraj/Downloads/missing_col1.cs
v",sep=",")
> data2
   V1       V2      V3         V4         V5
1   1     Rick  623.30 01/01/2012         IT
2   2      Dan  515.20 23/09/2013 Operations
3   3 Michelle  611.00 15/11/2014         IT
4   4     Ryan  729.00 11/05/2014         HR
5  NA     Gary  843.25 27/03/2015    Finance
6   6     Nina      NA 21/05/2013         IT
7   7    Simon  632.80 30/07/2013 Operations
8   8     Guru  722.50 17/06/2014    Finance
9   9     John      NA 21/05/2012           
10 10     Rock  600.80 30/07/2013         HR
11 11     Brad 1032.80 30/07/2013 Operations
12 12     Ryan  729.00 11/05/2014         HR
> data2 = read.csv(file ="C:/Users/Suraj/Downloads/missing_col1.csv",
col.names = c("Sno","Name","Salary","DataOfJoin","Department"))
> data2
   Sno     Name  Salary DataOfJoin Department
1    2      Dan  515.20 23/09/2013 Operations
2    3 Michelle  611.00 15/11/2014         IT
3    4     Ryan  729.00 11/05/2014         HR
4   NA     Gary  843.25 27/03/2015    Finance
5    6     Nina      NA 21/05/2013         IT
6    7    Simon  632.80 30/07/2013 Operations
7    8     Guru  722.50 17/06/2014    Finance
8    9     John      NA 21/05/2012           
9   10     Rock  600.80 30/07/2013         HR
10  11     Brad 1032.80 30/07/2013 Operations
11  12     Ryan  729.00 11/05/2014         HR
```

Error Detection and Correction

NA: Not Available - Known as missing values Works as a place holder for something that is ‘missing’. Most basic operations(addition, subtraction, multiplication, etc.) in R deal with it without crashing and return NA if one of the inputs is NA is.na(VALUE) is used to check if the input value is NA or not. Returns a TRUE/FALSE vector Whereas in case of Excel like utilities for numeric computations it’s assumed to be 0

```
> NA + 4
[1] NA
> V <- c(1,2,NA,3)
> median(V)
[1] NA
> median(V,na.rm=T)
[1] 2
> is.na(V)
[1] FALSE FALSE  TRUE FALSE
> naVals <- is.na(V)
> V[!naVals]
[1] 1 2 3
> V[complete.cases(V)]
[1] 1 2 3
> dataC <- read.csv(file="C:/Users/Suraj/Downloads/na_data.csv")
> dataC
   X1     Rick   X623.3 X01.01.2012         IT
1   2      Dan  515.20  23/09/2013 Operations
2   3 Michelle  611.00  15/11/2014         IT
3   4     Ryan  729.00  11/05/2014         HR
4  NA     Gary  843.25  27/03/2015    Finance
5   6     Nina      NA  21/05/2013         IT
6   7    Simon  632.80  30/07/2013 Operations
7   8     Guru  722.50  17/06/2014    Finance
8   9     John      NA  21/05/2012
9  10     Rock  600.80  30/07/2013         HR
10 11     Brad 1032.80  30/07/2013 Operations
11 12     Ryan  729.00  11/05/2014         HR
> dataCompleteCases <- dataC[complete.cases(dataC),]
> dataCompleteCases
   X1     Rick X623.3 X01.01.2012         IT
1   2      Dan  515.2  23/09/2013 Operations
2   3 Michelle  611.0  15/11/2014         IT
3   4     Ryan  729.0  11/05/2014         HR
6   7    Simon  632.8  30/07/2013 Operations
7   8     Guru  722.5  17/06/2014    Finance
9  10     Rock  600.8  30/07/2013         HR
10 11     Brad 1032.8  30/07/2013 Operations
11 12     Ryan  729.0  11/05/2014         HR
> 
```

Imputation

The process of estimating or deriving missing values, there are various methods for imputation

– Imputation of the mean

– Imputation of the median

– Imputation using linear regression models

• Package Hmisc implements many imputation methods, few examples:

Firstly, install Hmisc package.

```
> library(Hmisc)
Loading required package: lattice
Loading required package: survival
Loading required package: Formula
Loading required package: ggplot2

Attaching package: ‘Hmisc’

The following objects are masked from ‘package:dplyr’:

    src, summarize

The following objects are masked from ‘package:base’:

    format.pval, units

> x=c(1,2,3,NA,4,4,NA)

> x<-impute(x,fun=mean)
> x
   1    2    3    4    5    6    7 
 1.0  2.0  3.0 2.8*  4.0  4.0 2.8* 
> x<-impute(x,fun=median)
> x
   1    2    3    4    5    6    7 
 1.0  2.0  3.0 2.8*  4.0  4.0 2.8* 
>
```

Categorical Data:

Factors are variables in R which take on a limited number of different values; such variables are often referred to as categorical variables.

```
>
> gender_vector <- c("Male","Female","Male","Male")
> class(gender_vector)
[1] "character"
> factor_gender_vector <- factor(gender_vector)
> class(gender_vector)
[1] "character"
> class(factor_gender_vector)
[1] "factor"
> day_vector <- c('evening','morning','afternoon','midday','midnigh
t','evening','midnight')
> factor_day <- factor(day_vector, order = TRUE, levels=c('mornin
g','midday','afternoon','evening',midnight))
```

```
> gender <-c("male","male","transgender","female","male","female")
> employee <- data.frame(age,salary,gender)
> employee
  age salary      gender
1  40  10200        male
2  49  10230        male
3  48  12300 transgender
4  67  10444      female
5  52  12000        male
6  53  12333      female
>
> wfact = cut(employee$age,3, labels=c('Young','Medium','Aged'))
> table(wfact)
wfact
 Young Medium   Aged
     3      2      1
>
```

Conclusion:

I have successfully implemented and learned different data preprocessing techniques.