AIM: DATA PREPROCESSING IN R

THEORY:

Data Preprocessing Techniques

Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format.

Steps Involved in Data Preprocessing:

1. Data Cleaning:

The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

(a). Missing Data:

This situation arises when some data is missing in the data. It can be handled in various ways. Some of them are:

1. **Ignore the tuples:** This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.
2. **Fill the Missing values:** There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

(b). Noisy Data:

Noisy data is a meaningless data that can't be interpreted by machines.It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways :

1. **Binning Method:** This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segmented is handled separately. One can replace all data in a segment by its mean or boundary values can be used to complete the task.
2. **Regression:** Here data can be made smooth by fitting it to a regression function.The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).
3. **Clustering:** This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

2. Data Transformation:

This step is taken in order to transform the data in appropriate forms suitable for mining process. This involves following ways:

1. **Normalization :-** It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)

2. **Attribute Selection :-** In this strategy, new attributes are constructed from the given set of attributes to help the mining process.
3. **Discretization :-** This is done to replace the raw values of numeric attribute by interval levels or conceptual levels.
4. **Concept Hierarchy Generation :-** Here attributes are converted from level to higher level in hierarchy. For Example-The attribute "city" can be converted to "country".

3. Data Reduction:

Since data mining is a technique that is used to handle huge amount of data. While working with huge volume of data, analysis became harder in such cases. In order to get rid of this, we uses data reduction technique. It aims to increase the storage efficiency and reduce data storage and analysis costs.

The various steps to data reduction are:

1. **Data Cube Aggregation :-** Aggregation operation is applied to data for the construction of the data cube.
2. **Attribute Subset Selection :-** The highly relevant attributes should be used, rest all can be discarded. For performing attribute selection, one can use level of significance and p- value of the attribute.the attribute having p-value greater than significance level can be discarded.
3. **Numerosity Reduction :-** This enable to store the model of data instead of whole data, for example: Regression Models.
4. **Dimensionality Reduction :-** This reduce the size of data by encoding mechanisms.It can be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction are called lossless reduction else it is called lossy reduction. The two effective methods of dimensionality reduction are: Wavelet transforms and PCA (Principal Componenet Analysis).

A) PRINT HEAD OF MTCARS DATASET [PREDEFINED IN R]:

SOURCE CODE:

```
my_data<-mtcars
head(my_data,5)
```

OUTPUT:

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

B) READING DATA OF SPECFIC ROWS & COLUMNS:

SOURCE CODE:

```
#my_data
my_data1 <- my_data[1:6,1:5]
my_data1
```

OUTPUT:

```
> #my_data
> 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
```

C) RENAME COLUMN NAME USING DPLYR:

SOURCE CODE:

```
install.packages("dplyr")
library(dplyr, warn.conflicts = FALSE)
my_data1 = dplyr::rename(my_data1, "horse_power" = "hp")
my_data1
```

OUTPUT:

```
WARNING: Rtools is required to build R packages but is not currently installed. Plea
ll the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into ‘C:/Users/NARENDER KESWANI/Documents/R/win-library/4.0’
(as ‘lib’ is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/dplyr_1.0.8.zip'
Content type 'application/zip' length 1381575 bytes (1.3 MB)
downloaded 1.3 MB

package ‘dplyr’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\NARENDER KESWANI\AppData\Local\Temp\RtmpMTGJjs\downloaded_packages
> library(dplyr, warn.conflicts = FALSE)
Registered S3 methods overwritten by 'tibble':
  method     from
  format.tbl pillar
  print.tbl  pillar
Warning message:
package ‘dplyr’ was built under R version 4.0.5
> my_data1 = dplyr::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
```

D) ADDING NEW COLUMN:

SOURCE CODE:

```
## Adding new variable
my_data1$new_hp1 <- my_data1$horse_power * 0.5
colnames(my_data1)

my_data1
```

OUTPUT:

```
> ## Adding new variable
> my_data1$new_hp1 <- my_data1$horse_power * 0.5
> colnames(my_data1)
[1] "mpg"         "cyl"         "disp"        "horse_power" "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
```

E) READING DATA FROM CSV FILE:

SOURCE CODE:

```
#Reading with read.table() assumes no headers by default. First few lines :
data2 = read.table(file="C:\\Users\\NARENDER KESWANI\\Downloads\\missing_col1.csv",
sep = ",")
data2
```

OUTPUT:

```
> #Reading with read.table() assumes no headers by default. First few lines :
> data2 = read.table(file="C:\\Users\\NARENDER KESWANI\\Downloads\\missing_col1.csv", 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
```

F) READING DATA FROM SPECFIC COLUMNS OF CSV FILE:

SOURCE CODE:

```
#V1, V2, V3.. are given as default names (titles) by R
data2 = read.csv(file="C:\\Users\\NARENDER KESWANI\\Downloads\\missing_col1.csv",
col.names=c("Sno", "NAME","SALARY","DateOfJodata2"))
```

OUTPUT:

```
> #V1, V2, V3.. are given as default names (titles) by R
> data2 = read.csv(file="C:\\Users\\NARENDER KESWANI\\Downloads\\missing_col1.csv", col.names=c("Sno", "N
AME","SALARY","DateOfJodata2"))
Warning message:
In read.table(file = file, header = header, sep = sep, quote = quote,  :
  header and 'col.names' are of different lengths
> data2
        Sno    NAME     SALARY DateOfJodata2
2       Dan  515.20 23/09/2013    Operations
3  Michelle  611.00 15/11/2014            IT
4      Ryan  729.00 11/05/2014            HR
       Gary  843.25 27/03/2015       Finance
6      Nina      NA 21/05/2013            IT
7     Simon  632.80 30/07/2013    Operations
8      Guru  722.50 17/06/2014       Finance
9      John      NA 21/05/2012
10     Rock  600.80 30/07/2013            HR
11     Brad 1032.80 30/07/2013    Operations
12     Ryan  729.00 11/05/2014            HR
```

G) OPERATION WITH NA:

SOURCE CODE:

```
#Operation with NA
NA + 4
```

OUTPUT:

```
> #Operation with NA
> NA + 4
[1] NA
```

H) CREATE A VECTOR V WITH 1 NA VALUE:

SOURCE CODE:

```
#Create a vector V with 1 NA value
V <- c(1,2,NA,3)
V
```

OUTPUT:

```
> #Create a vector V with 1 NA value
> V <- c(1,2,NA,3)
> V
[1]  1  2 NA  3
```

I) FIND MEDIAN:

1) WITH NA:

SOURCE CODE:

```
#Median with NA
median(V)
```

OUTPUT:

```
> #Median with and without NA (remove NA)
> #with NA
> median(V)
[1] NA
```

2) WITHOUT NA:

SOURCE CODE:

```
#On removing NAs
median(V, na.rm = T)
```

OUTPUT:

```
> #without NA
> #On removing NAs
> median(V, na.rm = T)
[1] 2
```

J) **CHECK WHETHER IT IS NA OR NOT:**

SOURCE CODE:

```
#Apply is.na() to vector
is.na(V)
```

OUTPUT:

```
> #Apply is.na() to vector
> is.na(V)
[1] FALSE FALSE  TRUE FALSE
```

K) **REMOVING THE NA VALUES BY USING LOGICAL INDEXING:**

SOURCE CODE:

```
#Removing the NA values by using logical indexing
naVals <- is.na(V)
```

OUTPUT:

```
> #Removing the NA values by using logical indexing
> naVals <- is.na(V)
> naVals
[1] FALSE FALSE  TRUE FALSE
```

L) **Get values that are not NA**

SOURCE CODE:

```
V[!naVals]
```

OUTPUT:

```
> #Get values that are not NA
> V[!naVals]
[1] 1 2 3
```

M) SUBSETTING WITH COMPLETE CASES - VALUES THAT ARE NOT NA:

SOURCE CODE:

```
#Subsetting with complete cases - values that are not NA
V[complete.cases(V)]
```

OUTPUT:

```
> #Subsetting with complete cases - values that are not NA
> V[complete.cases(V)]
[1] 1 2 3
```

N) SUBSETTING A DATA FRAME WITH COMPLETE CASES:

SOURCE CODE:

```
#Subsetting a data frame with complete cases
#Complete Data of Prime Ministers. Notice NAs
dataC <- read.csv(file ="C:\\Users\\NARENDER KESWANI\\Downloads\\na_data.csv",
na.strings = "")
dataC

# Subset only the rows without NA
dataCompleteCases <- dataC[complete.cases(dataC),]
dataCompleteCases
```

OUTPUT:

```
> #Subsetting a data frame with complete cases
> #Complete Data of Prime Ministers. Notice NAs
> dataC <- read.csv(file ="C:\\Users\\NARENDER KESWANI\\Downloads\\na_data.csv", na.strings = "")
> 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       <NA>
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
```

```
> # Subset only the rows without NA
> 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
```

O) MEAN IMPUTATION & MEDIAN IMPUTATION

SOURCE CODE:

```
install.packages('Hmisc')
library(Hmisc)

## create a vector
x = c(1,2,3,NA,4,4,NA)
x

# mean imputation - from package, mention name of function to be used
x <- impute(x, fun = mean)
x

#median imputation
x <- impute(x, fun = median)
x
```

OUTPUT:

```
> ## create a vector
> x = c(1,2,3,NA,4,4,NA)
> # mean imputation - from package, mention name of function to be used
> 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* 
> 
> #median imputation
> 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* 
```

P) CONVERT:

1) Convert Character into Factor(categorical data):

SOURCE CODE:

```
#Convert Character into Factor(categorical data)
#Create gender vector
gender_vector <- c("Male", "Female", "Female", "Male", "Male")
class(gender_vector)

#Convert gender_vector to a factor
factor_gender_vector <-factor(gender_vector)
class(factor_gender_vector)
```

OUTPUT:

```
> #Convert Character into Factor(categorical data)
> #Create gender vector
> gender_vector <- c("Male", "Female", "Female", "Male", "Male")
> class(gender_vector)
[1] "character"
>
> #Convert gender_vector to a factor
> factor_gender_vector <-factor(gender_vector)
> class(factor_gender_vector)
[1] "factor"
```

Q) CREATE ORDINAL CATEGORICAL VECTOR:

SOURCE CODE:

```
#Create Ordinal categorical vector
day_vector <- c('evening', 'morning', 'afternoon', 'midday', 'midnight', 'evening')
day_vector
```

OUTPUT:

```
> #Create Ordinal categorical vector
> day_vector <- c('evening', 'morning', 'afternoon', 'midday', 'midnight', 'evening')
> day_vector
[1] "evening"   "morning"   "afternoon" "midday"    "midnight"  "evening"
```

R) CONVERT VECTOR INTO A FACTOR WITH ORDERED LEVEL:

SOURCE CODE:

```
#Convert `day_vector` to a factor with ordered level
factor_day <- factor(day_vector, order = TRUE, levels =c('morning', 'midday', 'afternoon',
'evening', 'midnight'))

#Print the new variable
factor_day
```

OUTPUT:

```
> #Convert `day_vector` to a factor with ordered level
> factor_day <- factor(day_vector, order = TRUE, levels =c('morning', 'midday', 'afternoon', 'evening',
 'midnight'))
> #Print the new variable
> factor_day
[1] evening   morning   afternoon midday    midnight  evening
Levels: morning < midday < afternoon < evening < midnight
```

S) CREATE DATAFRAME FROM VECTOR:

SOURCE CODE:

```
# Creating vectors
age <- c(40, 49, 48, 40, 67, 52, 53)
salary <- c(103200, 106200, 150200, 10606, 10390, 14070, 10220)
gender <- c("male", "male", "transgender","female", "male", "female", "transgender")

# Creating data frame named employee
employee<- data.frame(age, salary, gender)
employee
```

OUTPUT:

```
> # Creating vectors
> age <- c(40, 49, 48, 40, 67, 52, 53)
> salary <- c(103200, 106200, 150200, 10606, 10390, 14070, 10220)
> gender <- c("male", "male", "transgender","female", "male", "female", "transgender")
> # Creating data frame named employee
> employee<- data.frame(age, salary, gender)
> employee
  age salary      gender
1  40 103200        male
2  49 106200        male
3  48 150200 transgender
4  40  10606      female
5  67  10390        male
6  52  14070      female
7  53  10220 transgender
```

T) CERATE FACTOR WITH LABELS:

SOURCE CODE:

```
# Creating a factor corresponding to age with labels
wfact = cut(employee$age, 3, labels=c('Young', 'Medium', 'Aged'))
table(wfact)
```

OUTPUT:

```
> # Creating a factor corresponding to age with labels
> wfact = cut(employee$age, 3, labels=c('Young', 'Medium', 'Aged'))
> table(wfact)
wfact
 Young Medium   Aged
     4      2      1
```

CONCLUSION:

From this practical, I have learned data preprocessing techniques in R.