2

DATA PREPROCESSING

Implementation in R

In the previous article, I discussed about data preprocessing and the theoretical explanation behind it. In this article, the focus will be on implementing the complete data preprocessing step in R programming Language.

IMPORTING THE DATASET

In R, we can import our dataset using the read.csv() function in the following manner.

```
dataset = read.csv('file-name.csv')
```

It is important to note that in R, we don't need to distinguish between Dependent and Independent variable vector. (Independent variable vector is commonly called Matrix of Features as these features of the data define the dependent variable)

TAKING CARE OF MISSING DATA

As we know that missing data is taken care by filling the missing value with mean values (mostly).The ifelse(test , yes , no) , conditional element selection statement is used here.

```
dataset$column-name = ifelse(is.na(dataset$column-name),
                     ave(dataset$column-name, FUN = function(x) mean(x, na.rm = TRUE)),
                     dataset$column-name)
```

is.na() checks if the value is missing, it takes column name as parameter, if it returns yes, then statement at second parameter is processed, otherwise the third statement is processed.

ave(x, ..., FUN = mean) is a function that returns group averages over level combination of factors. In other way it returns the average of data of similar kind.

The third parameter is left as column name itself because if there is no missing value, then the actual column content should be retained.

So, this is how we take care of Missing Data.

CATEGORICAL DATA

Categorical Data is taken care by encoding data in form of factors (the term 'category' or 'enumerated data' is also used for factors). One can understand it as assigning values to the data to make it significant to be used in a mathematical model. Here is an example.

```
dataset$Purchased = factor(dataset$Purchased,
                           levels = c('No','Yes'),
                           labels = c(0,1))
```

In the above example, if the value in column Purchased is 'Yes' it gets encoded by a factor 1 and 'No' is encoded by 0.

SPLITTING DATA INTO TRAINING & TEST SET

This is the most important part as our machine learning model heavily depends upon how well it is trained by the Training Set. It is done as follows

```
split = sample.split(dataset$Dependent-variable-column, SplitRatio = 0.8)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
```

Firstly, split the data w.r.t. dependent variable and the split ratio is the training set ratio (ratio of entries from the data that are to be present in Training Set) Obviously, training set ratio + test set ratio = 1.

Afterwards, declare training set and test set as the subsets of data (which they actually are). For Training set, split is always TRUE and vice versa.

FEATURE SCALING

It is the simplest, yet it produces a headache.
We simply have to use scale() function, but we need to take care that the data it works upon is numeric.

```
training_set[, 2:3] = scale(training_set[, 2:3])
test_set[, 2:3] = scale(test_set[, 2:3])
```

Factors that we assigned to our categorical data are not numeric in R. Thus to avoid the possible errors we may encounter, we should specify the column on which scaling has to be done. In above example, 2:3 is nothing but column number 2 and 3.

This completes the Data Preprocessing part in R.