Data preprocessing

Frarlon Rodriguez

5/25/2019

Introduction

In this tutorial, you will learn about data preprocessing. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

You'll learn to handle missing data, categorical data, divide the data into train and test and feature scaling.

Importing the dataset

The first step is to import the dataset, then read it using the read function.

```
dataset = read.csv('Data.csv')

dataset
```

```
##    Country Age Salary Purchased
## 1   France  44  72000        No
## 2    Spain  27  48000       Yes
## 3  Germany  30  54000        No
## 4    Spain  38  61000        No
## 5  Germany  40     NA       Yes
## 6   France  35  58000       Yes
## 7    Spain  NA  52000        No
## 8   France  48  79000       Yes
## 9  Germany  50  83000        No
## 10  France  37  67000       Yes
```

Missing Data

The data is about if someone purchased an item or not. The columns We have are; country, age, salary and Purchased.

Now let's perform a summary in the data to see more information.

```
summary(dataset)
```

```
##     Country       Age             Salary      Purchased
##  France :4   Min.   :27.00   Min.   :48000   No :5
##  Germany:3   1st Qu.:35.00   1st Qu.:54000   Yes:5
##  Spain  :3   Median :38.00   Median :61000
##              Mean   :38.78   Mean   :63778
##              3rd Qu.:44.00   3rd Qu.:72000
##              Max.   :50.00   Max.   :83000
##              NA's   :1       NA's   :1
```

We can clearly see from the summary that we have salary and age missing. One missing data in Germany that is salary and another in Spain that is age.

So, how can we handle this problem?

- Remove the rows that have missing data.
- We can use the mean of the entire data to replace the missing data.
- Predict the missing Values.

In this case let's take the second approach, in case you want more information about the pros and cons of different approaches from handling missing data refer to https://www.analyticsindiamag.com/5-ways-handle-missing-values-machine-learning-datasets/ (https://www.analyticsindiamag.com/5-ways-handle-missing-values-machine-learning-datasets/).

The following is the way to do it.

```
dataset$Age = ifelse(is.na(dataset$Age),
                     ave(dataset$Age, FUN = function(x) mean(x, na.rm = TRUE)),
                     dataset$Age)
dataset$Salary = ifelse(is.na(dataset$Salary),
                        ave(dataset$Salary, FUN = function(x) mean(x, na.rm = TRUE)),
                        dataset$Salary)
```

- The *ifelse* will receive 3 params, as you can see the first one is the condition, the second if the condition is true and then the third one if the condition is false. So the condition means If have missing data for this column then do the mean of the entire age column and then replace the missing data for the calculated mean
- The second for salary is the same approach, we have the condition if we have missing data for the salary column the compute the mean for the salary.

Now Let's see the data.

```
dataset
```

```
##    Country      Age   Salary Purchased
## 1   France 44.00000 72000.00        No
## 2    Spain 27.00000 48000.00       Yes
## 3  Germany 30.00000 54000.00        No
## 4    Spain 38.00000 61000.00        No
## 5  Germany 40.00000 63777.78       Yes
## 6   France 35.00000 58000.00       Yes
## 7    Spain 38.77778 52000.00        No
## 8   France 48.00000 79000.00       Yes
## 9  Germany 50.00000 83000.00        No
## 10  France 37.00000 67000.00       Yes
```

You can see that the missing data has been replaced by the average of each category.

Categorical Data

The following is the definition of categorical variables. A categorical variable (sometimes called a nominal variable) is one that has two or more categories, but there is no intrinsic ordering to the categories. For example, gender is a categorical variable having two categories (male and female) and there is no intrinsic ordering to the categories.

Categorical variables are important to be encoded since machine learning models are based on mathematical equations and text won't make sense so some transformation of the categorical variables has to be done.

```
dataset
```

```
##    Country      Age   Salary Purchased
## 1   France 44.00000 72000.00        No
## 2    Spain 27.00000 48000.00       Yes
## 3  Germany 30.00000 54000.00        No
## 4    Spain 38.00000 61000.00        No
## 5  Germany 40.00000 63777.78       Yes
## 6   France 35.00000 58000.00       Yes
## 7    Spain 38.77778 52000.00        No
## 8   France 48.00000 79000.00       Yes
## 9  Germany 50.00000 83000.00        No
## 10  France 37.00000 67000.00       Yes
```

So in our dataset, we can clearly see that we two categorical variables, country and purchased. Now let's transform them into something machine learning models can understand.

```
dataset$Country = factor(dataset$Country,
                         levels = c('France','Spain','Germany'),
                         labels = c(1, 2, 3))
```

Now let's do the same for purchase

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

See the dataset and see how the new values are there for each variable

```
dataset
```

```
##    Country      Age   Salary Purchased
## 1        1 44.00000 72000.00         0
## 2        2 27.00000 48000.00         1
## 3        3 30.00000 54000.00         0
## 4        2 38.00000 61000.00         0
## 5        3 40.00000 63777.78         1
## 6        1 35.00000 58000.00         1
## 7        2 38.77778 52000.00         0
## 8        1 48.00000 79000.00         1
## 9        3 50.00000 83000.00         0
## 10       1 37.00000 67000.00         1
```

Split data set

We need to split the dataset in training set and test set, but why we need to do this?. We need to do this because machine learning models will learn from our data and make predictions but we need always need a new set to make sure it will work and test the performance and how they adapt to a new set and new situations.

```
library(caTools)
set.seed(123)
split = sample.split(dataset$Purchased, SplitRatio = 0.8)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
```

We start by importing a library that will help us to split our data. Then we use a seed this is to make sure the same random result of splitting data can be reproduced each time.

- A training set that you can use to train your model and find optimal parameters
- A test set that you can use to test your trained model and see how well it functions.

You can read more about it here. https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7 (https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7)

The split radio people usually tend to start with a 80-20% split (80% training set – 20% test set). It's usually a good start but it's more a rule of thumb than anything else and you may want to adjust the splits depending on the amount of available data.

The key principle to understand is that the more samples the lower the variance. So you need the training set to be big enough to achieve low variance over the model parameters.

https://www.beyondthelines.net/machine-learning/how-to-split-a-dataset/ (https://www.beyondthelines.net/machine-learning/how-to-split-a-dataset/)

Feature Scaling

Most of the times, your dataset will contain features highly varying in magnitudes, units, and range. But since most of the machine learning algorithms use Euclidean distance between two data points in their computations, this is a problem.

If left alone, these algorithms only take in the magnitude of features neglecting the units. The results would vary greatly between different units, 5kg, and 5000gms. The features with high magnitudes will weigh in a lot more in the distance calculations than features with low magnitudes.

To suppress this effect, we need to bring all features to the same level of magnitudes. This can be achieved by scaling.

https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e (https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e)

As in the example, we are working on, We need to scale salary and age. How we do this in R? well is actually pretty simple

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

Now let's see our data

```
training_set
```

```
##    Country         Age     Salary Purchased
## 1        1  0.90101716  0.9392746         0
## 2        2 -1.58847494 -1.3371160         1
## 3        3 -1.14915281 -0.7680183         0
## 4        2  0.02237289 -0.1040711         0
## 5        3  0.31525431  0.1594000         1
## 7        2  0.13627122 -0.9577176         0
## 8        1  1.48678000  1.6032218         1
## 10       1 -0.12406783  0.4650265         1
```

```
test_set
```

```
##   Country        Age     Salary Purchased
## 6       1 -0.7071068 -0.7071068         1
## 9       3  0.7071068  0.7071068         0
```

You can see how our data is scaled(age and salary) now our data is ready for a machine learning model. Something important is that feature scaling is done by some algorithms, but be aware that some don't do it and you'll need to do it.

Now is your turn to do some data preprocessing.