Automating Data Exploration with R

Imputing Missing Data

You could write a whole book on the topic of dealing with missing data. Here, we'll aim to cover the most typical cases and write-off exceptions to be customized by hand. `NA` s can mean different things in different contexts. The two types of imputation we'll look at are replacing them with `0` 's and replacing them with mean value of that feature. Before any imputation, we will create a new binary feature stating whether a cell was an `NA` or not.

When should you impute with 0 versus its mean value? It all depends on the context. Imagine a library count of books checked out - if a patron has `NA` , you wouldn't want to replace it with the mean value for that field, but a `0` would fit nicely. If the patron has never checked out a book, then they have 0 books checked out. On the other hand, imagine a feature holding the age of customers, you wouldn't want to replace NA's with 0's as a customer with no reported age will never be 0 years of age. Here, taking the mean value of all ages recorded works well. Unfortunately, there are no easy rules. From a modeling perspective, taking the mean is a relatively easy way to get those turned into numbers without affecting the model too drastically.

Let's build a simple data set to wrap our heads around these questions:

```
mix_dataset <- data.frame(
               id=c(1,NA,3,4,5),
               mood=c(0,20,20,Inf,50),
               value=c(12.34, 32.2, NaN, 83.1, 8.32),
               outcome=c(1,1,0,0,0))
head(mix_dataset)
```

```
##   id mood value outcome
## 1  1    0 12.34       1
## 2 NA   20 32.20       1
## 3  3   20   NaN       0
## 4  4  Inf 83.10       0
## 5  5   50  8.32       0
```

So, how do we find these NA's programmatically?

```
mix_dataset_temp <- mix_dataset

# where are the NAs?
is.na(mix_dataset_temp)
```

```
##         id  mood value outcome
## [1,] FALSE FALSE FALSE   FALSE
## [2,]  TRUE FALSE FALSE   FALSE
## [3,] FALSE FALSE  TRUE   FALSE
## [4,] FALSE FALSE FALSE   FALSE
## [5,] FALSE FALSE FALSE   FALSE
```

In many situations, you'll want to customize the imputation of your feature. Here we'll go over imputing with 0 or mean, but you could use a custom number or the max, min, median, etc.

```
# impute column:
mix_dataset_temp$id[is.na(mix_dataset_temp$id)] <- 0
mix_dataset_temp
```

```
##   id mood value outcome
## 1  1    0 12.34       1
## 2  0   20 32.20       1
## 3  3   20   NaN       0
## 4  4  Inf 83.10       0
## 5  5   50  8.32       0
```

```
# impute with mean
mix_dataset_temp$value[is.nan(mix_dataset_temp$value)] <- mean(mix_dataset_temp$va
lue, na.rm = TRUE)
mix_dataset_temp
```

```
##   id mood value outcome
## 1  1    0 12.34       1
## 2  0   20 32.20       1
## 3  3   20 33.99       0
## 4  4  Inf 83.10       0
## 5  5   50  8.32       0
```

It may be useful for your model to mark all the `NA` , `inf` , `NaN` before removing them. We make that an optional parameter that is turned off by default. We also add a parameter to prune out any feature with zero variance, where all the values are the same. We are making it optional as there may be some cases where you may want to see it while exploring, but this won't help (and may even hurt) your modeling efforts.

```r
Impute_Features <- function(data_set, features_to_ignore=c(),
                            use_mean_instead_of_0=TRUE,
                            mark_NAs=FALSE,
                            remove_zero_variance=FALSE) {

     for (feature_name in setdiff(names(data_set), features_to_ignore)) {
          print(feature_name)
          # remove any fields with zero variance
          if (remove_zero_variance) {
               if (length(unique(data_set[, feature_name]))==1) {
                    data_set[, feature_name] <- NULL
                    next
               }
          }

          if (mark_NAs) {
               # note each field that contains missing or bad data
               if (any(is.na(data_set[,feature_name]))) {
                    # create binary column before imputing
                    newName <- paste0(feature_name, '_NA')
                    data_set[,newName] <- as.integer(ifelse(is.na(data_set[,featur
e_name]),1,0)) }

               if (any(is.infinite(data_set[,feature_name]))) {
                    newName <- paste0(feature_name, '_inf')
                    data_set[,newName] <- as.integer(ifelse(is.infinite(data_set[,
feature_name]),1,0)) }
          }

          if (use_mean_instead_of_0) {
               data_set[is.infinite(data_set[,feature_name]),feature_name] <- NA
               data_set[is.na(data_set[,feature_name]),feature_name] <- mean(data_
set[,feature_name], na.rm=TRUE)

          } else {
               data_set[is.na(data_set[,feature_name]),feature_name] <- 0
               data_set[is.infinite(data_set[,feature_name]),feature_name] <- 0
          }
     }
     return(data_set)
}


mix_dataset_temp <- Impute_Features(mix_dataset, use_mean_instead_of_0 = TRUE, mar
k_NAs = TRUE)
```

```
## [1] "id"
## [1] "mood"
## [1] "value"
## [1] "outcome"
```

```
head(mix_dataset_temp)
```

```
##     id mood value outcome id_NA mood_inf value_NA
## 1 1.00  0.0 12.34       1     0        0        0
## 2 3.25 20.0 32.20       1     1        0        0
## 3 3.00 20.0 33.99       0     0        0        1
## 4 4.00 22.5 83.10       0     0        1        0
## 5 5.00 50.0  8.32       0     0        0        0
```