Automating Data Exploration with R

Integers

First thing we have to do to extract additional intelligence out of an integer is to verify that it actually is an integer:

```
print(is.integer(1))
```

```
## [1] FALSE
```

```
print(class(1))
```

```
## [1] "numeric"
```

```
print(class(1L))
```

```
## [1] "integer"
```

We can't count on the `is.integer` function as it requires the value to be declared as an integer literal (L). Instead we'll use the round function (in R 3.3.3 and up you can use `is.wholenumber`). We'll explore some simple feature engineering to capture:

- Is feature equal to zero
- Is feature positive
- Binning feature values

```
mix_dataset <- data.frame(
               id=c(1,2,3,4,5),
               mood=c(0,20,20,40,50),
               value=c(12.34, 32.2, 24.3, 83.1, 8.32),
               outcome=c(1,1,0,0,0))

library(readr)
write_csv(mix_dataset, 'mix_dataset.csv')
mix_dataset <- read_csv('mix_dataset.csv')

Feature_Engineer_Integers <- function(data_set, features_to_ignore=c()) {
     require(infotheo)
     data_set <- data.frame(data_set)

     for (feature_name in setdiff(names(data_set), features_to_ignore)) {
          if (class(data_set[,feature_name])=='numeric' | class(data_set[,feature_
name])=='integer') {
               feature_vector <- data_set[,feature_name]

               if (all((feature_vector - round(feature_vector)) == 0)) {
                    # make sure we have more than 2 values excluding NAs
                    if (length(unique(data_set[,feature_name][!is.na(data_set[,fea
ture_name])])) > 2) {
                         print(feature_name)
                         data_set[,paste0(feature_name,'_IsZero')] <- ifelse(data_
set[,feature_name]==0,1,0)
                         data_set[,paste0(feature_name,'_IsPositive')] <- ifelse(d
ata_set[,feature_name]>=0,1,0)
                         # separate data into two bins
                         data_discretized <- discretize(data_set[,feature_name], d
isc='equalfreq', nbins=2)
                         data_set[,paste0(feature_name,'_2Bins')] <- data_discreti
zed$X

                         if (length(unique(data_set[,feature_name][!is.na(data_set
[,feature_name])])) > 4) {
                              # try 4 bins
                              data_discretized <- discretize(data_set[,feature_nam
e], disc='equalfreq', nbins=4)
                              data_set[,paste0(feature_name,'_4Bins')] <- data_dis
cretized$X
                         }
                    }
               }
          }
     }
     return (data_set)
}

mix_dataset <- read_csv('mix_dataset.csv')
```

```
Feature_Engineer_Integers(mix_dataset, features_to_ignore=c('id'))
```

```
## Loading required package: infotheo
```

```
## [1] "mood"
```

```
##   id mood value outcome mood_IsZero mood_IsPositive mood_2Bins
## 1  1    0 12.34       1           1               1          1
## 2  2   20 32.20       1           0               1          1
## 3  3   20 24.30       0           0               1          1
## 4  4   40 83.10       0           0               1          2
## 5  5   50  8.32       0           0               1          2
```

Numbers

Feature engineering of numbers is an enormous subject that we'll keep under control here. Most feature engineering should come out of the business context, something we can automate here. Here, we'll look at some simple transformations that are applicable to a lot of data sets on whole/real numbers:

```r
Feature_Engineer_Numbers <- function(data_set, features_to_ignore=c()) {
     require(infotheo)
     data_set <- data.frame(data_set)
     date_features <- setdiff(names(data_set[sapply(data_set, is.numeric)]), features_to_ignore)
     for (feature_name in date_features) {
          feature_vector <- data_set[,feature_name]
          if (is.integer(feature_vector) | is.numeric(feature_vector)) {
               if (any((feature_vector - round(feature_vector)) != 0)) {
                    # make sure we have more than 2 values excluding NAs
                    if (length(unique(data_set[,feature_name][!is.na(data_set[,feature_name])])) > 2) {
                         print(feature_name)
                         # polynomial transformation
                         poly_vector <- poly(x=feature_vector, degree = 2)
                         data_set[,paste0(feature_name, "_poly1")] <- poly_vector[,1]
                         data_set[,paste0(feature_name, "_poly2")] <- poly_vector[,2]
                         # log transform
                         data_set[,paste0(feature_name, "_log")] <- log(x = feature_vector)
                         # exponential transform
                         data_set[,paste0(feature_name, "_exp")] <- exp(x = feature_vector)
                         # rounding
                         data_set[,paste0(feature_name, "_rnd")] <- round(x = feature_vector, digits = 0)
                         # binning into 2 bins
                         data_discretized <- discretize(data_set[,feature_name], disc='equalfreq', nbins=2)
                         data_set[,paste0(feature_name,'_2Bins')] <- data_discretized$X

                    }
               }
          }
     }
     return(data_set)
}

mix_dataset <- data.frame(
               id=sample(1:100, 100, replace=F),
               value=runif(100, 1.0, 55.5)
)
write_csv(mix_dataset, 'mix_dataset.csv')
mix_dataset <- read_csv('mix_dataset.csv')
head(Feature_Engineer_Numbers(mix_dataset, features_to_ignore=c()))
```

```
## [1] "value"
```

```
##   id     value  value_poly1 value_poly2 value_log    value_exp value_rnd
## 1 87 19.974386 -0.043198041 -0.07035281 2.9944507 4.728959e+08        20
## 2 98 51.824357  0.184762399  0.19461439 3.9478603 3.213900e+22        52
## 3 25 18.063778 -0.056872870 -0.05327673 2.8939087 6.998408e+07        18
## 4  4  2.469639 -0.168485118  0.22562557 0.9040719 1.181818e+01         2
## 5 53 31.098497  0.036420785 -0.09565801 3.4371595 3.205574e+13        31
## 6 51 26.661576  0.004664319 -0.10073098 3.2832234 3.792934e+11        27
##   value_2Bins
## 1           1
## 2           2
## 3           1
## 4           1
## 5           2
## 6           2
```

The `Feature_Engineer_Numbers` function will only transform features containing real numbers. It then applies a 2-degree polynomial transform, a simple log and exponential transform. It also rounds the data and splits it into two buckets using library `infotheo`. All these transformations are highly customizable, you could try 3-degree polynomial transform, round only to the 1st or 2nd digit. You could split the data into many more bins. Depending on the data, a few things can break, whether or not you have negative numbers, and too much or too little variation. This would definitely be an ideal candidate for a `try` / `catch` error handling.

For more information on the following transformer functions:

- Log, Exp (http://www.inside-r.org/r-doc/base/log)
- poly/log (https://stat.ethz.ch/R-manual/R-devel/library/stats/html/poly.html)
- discretize (http://www.inside-r.org/packages/cran/infotheo/docs/discretize)