Automating Data Exploration with R

Pipeline Check

Before we move into feature engineering, let's do a test run through all our pipeline functions we've built so far. We'll build a test data set with interesting data for our functions to clean up.

Let's load our pipeline functions:

```
Get_Free_Text_Measures <- function(data_set, minimum_unique_threshold=0.9, feature
s_to_ignore=c()) {

     # look for text entries that are mostly unique
     text_features <- c(names(data_set[sapply(data_set, is.character)]), names(dat
a_set[sapply(data_set, is.factor)]))
     for (f_name in setdiff(text_features, features_to_ignore)) {
          f_vector <- as.character(data_set[,f_name])

          # treat as raw text if data over minimum_precent_unique unique
          if (length(unique(as.character(f_vector))) > (nrow(data_set) * minimum_u
nique_threshold)) {
               data_set[,paste0(f_name, '_word_count')] <- sapply(strsplit(f_vecto
r, " "), length)
               data_set[,paste0(f_name, '_character_count')]  <-  nchar(as.charact
er(f_vector))
               data_set[,paste0(f_name, '_first_word')] <- sapply(strsplit(as.char
acter(f_vector), " "), `[`, 1)
               # remove orginal field
               data_set[,f_name] <- NULL
          }
     }
     return(data_set)
}
```

```
Binarize_Features <- function(data_set, features_to_ignore=c(), leave_out_one_leve
l=FALSE, max_level_count=20) {
     require(dplyr)
     text_features <- c(names(data_set[sapply(data_set, is.character)]), names(dat
a_set[sapply(data_set, is.factor)]))
     for (feature_name in setdiff(text_features, features_to_ignore)) {
          feature_vector <- as.character(data_set[,feature_name])

          # check that data has more than one level
          if (length(unique(feature_vector)) == 1)
               next

          # We set any non-data to text
          feature_vector[is.na(feature_vector)] <- 'NA'
          feature_vector[is.infinite(feature_vector)] <- 'INF'
          feature_vector[is.nan(feature_vector)] <- 'NAN'

          # only give us the top x most popular categories
          temp_vect <- data.frame(table(feature_vector)) %>% arrange(desc(Freq)) %
>% head(max_level_count)
          feature_vector <- ifelse(feature_vector %in% temp_vect$feature_vector, f
eature_vector, 'Other')

          # loop through each level of a feature and create a new column
          first_level=TRUE
          for (newcol in unique(feature_vector)) {
               if (leave_out_one_level & first_level) {
                    # avoid dummy trap and skip first level
                    first_level=FALSE
                    next
               }

               data_set[,paste0(feature_name,"_",newcol)] <- ifelse(feature_vector
==newcol,1,0)
          }
          # remove original feature
          data_set <- data_set[,setdiff(names(data_set),feature_name)]
     }
     return (data_set)
}
```

```
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 <- data.frame(
               ids=c(1,NA,3,4,5),
               some_dates = c('01/11/2012','04/12/2012','28/02/2013','17/06/201
4','08/03/2015'),
               mood=c(0,20,20,Inf,50),
               some_real_numbers = c(12.34, 32.2, NaN, 83.1, 8.32),
               some_text = c('sentence one','sentence two', 'mixing it up', 'sente
nce four', 'sentence five'))
head(mix_dataset)
```

```
##   ids some_dates mood some_real_numbers     some_text
## 1   1 01/11/2012    0             12.34  sentence one
## 2  NA 04/12/2012   20             32.20  sentence two
## 3   3 28/02/2013   20               NaN  mixing it up
## 4   4 17/06/2014  Inf             83.10 sentence four
## 5   5 08/03/2015   50              8.32 sentence five
```

```
library(readr)
write_csv(mix_dataset, 'mix_dataset.csv')
```

```
# take a peek at the data
readLines('mix_dataset.csv', n=3)
```

```
## [1] "ids,some_dates,mood,some_real_numbers,some_text"
## [2] "1,01/11/2012,0,12.34,sentence one"
## [3] "NA,04/12/2012,20,32.2,sentence two"
```

```
# pick your reader
library(data.table)
mix_dataset <- fread('mix_dataset.csv', data.table = FALSE)

# format date field to be R compliant
mix_dataset$some_dates <-  as.Date(mix_dataset$some_dates, format="%d/%m/%Y")
str(mix_dataset$some_dates)
```

```
##  Date[1:5], format: "2012-11-01" "2012-12-04" "2013-02-28" "2014-06-17" ...
```

```
# extra quantative value out of text entires
mix_dataset <- Get_Free_Text_Measures(data_set = mix_dataset)
head(mix_dataset,2)
```

```
##   ids some_dates mood some_real_numbers some_text_word_count
## 1   1 2012-11-01    0             12.34                    2
## 2  NA 2012-12-04   20             32.20                    2
##   some_text_character_count some_text_first_word
## 1                        12             sentence
## 2                        12             sentence
```

```
# binarize categories
mix_dataset <- Binarize_Features(data_set = mix_dataset, features_to_ignore = c(),
leave_out_one_level = TRUE)
```

```
## Loading required package: dplyr
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:data.table':
##
##     between, last
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
head(mix_dataset, 2)
```

```
##   ids some_dates mood some_real_numbers some_text_word_count
## 1   1 2012-11-01    0             12.34                    2
## 2  NA 2012-12-04   20             32.20                    2
##   some_text_character_count some_text_first_word_mixing
## 1                        12                           0
## 2                        12                           0
```

```
# impute missing data using 0
mix_dataset <- Impute_Features(mix_dataset, use_mean_instead_of_0 = FALSE, feature
s_to_ignore = c('some_dates'))
```

```
## [1] "ids"
## [1] "mood"
## [1] "some_real_numbers"
## [1] "some_text_word_count"
## [1] "some_text_character_count"
## [1] "some_text_first_word_mixing"
```

```
mix_dataset
```

```
##   ids some_dates mood some_real_numbers some_text_word_count
## 1   1 2012-11-01    0             12.34                    2
## 2   0 2012-12-04   20             32.20                    2
## 3   3 2013-02-28   20              0.00                    3
## 4   4 2014-06-17    0             83.10                    2
## 5   5 2015-03-08   50              8.32                    2
##   some_text_character_count some_text_first_word_mixing
## 1                        12                           0
## 2                        12                           0
## 3                        12                           1
## 4                        13                           0
## 5                        13                           0
```