Homework 4

Due: 11.59pm on Friday, February 14

Submission instructions:

- Submit one write-up per group on gradescope.com. Please do not bring printouts of your solutions to the classroom.

Question 1

In this homework, you will be analyzing the data coming from KDD Cup 2009: Customer relationship prediction. The KDD Cup is a annual data mining competition and the data were given by French Telecom company Orange. There are 3 (binary) response variables to explore:

- churn: predicting the propensity of customers to switch provider
- appetency: predicting the propensity of customers to buy new products or services
- up-selling: predicting the propensity of customers to buy upgrades or add-ons proposed to them to make the sale more profitable

We are using the small dataset consists of 50,000 observations and 230 explanatory variables (the large one comes with 15,000 explanatory variables). The first 190 variables are numerical and the last 40 are categorical. Due to the nature of competition and privacy concerns, none of the explanatory variables come with meaningful names, are the categorical variables are encoded anonymously as well.

1. Getting the Data

Data are located in the repository here:
https://github.com/ChicagoBoothML/MLClassData/tree/master/KDDCup2009_Customer_relationship

The "orange_small_train.data.gz" contains the explanatory variables. We load it in R as X. There are three txt files storing the response variables, we code them as `Y_churn`, `Y_appetency` and `Y_upselling`.

The following code will help you load it into R.

```
# Check if needed libraries need are installed, if not install them.
if("R.utils" %in% rownames(installed.packages()) == FALSE) {
  install.packages("R.utils")
  }
if("data.table" %in% rownames(installed.packages()) == FALSE) {
  install.packages("data.table")
  }

#load libraries
library("R.utils")
library("data.table")

# Download data
gitURL =
  "https://github.com/ChicagoBoothML/MLClassData/raw/master/KDDCup2009_Customer_relationship/";
DownloadFileList = c("orange_small_train.data.gz",
                     "orange_small_train_appetency.labels.txt",
                     "orange_small_train_churn.labels.txt",
                     "orange_small_train_upselling.labels.txt")
```

```
LoadFileList=c("orange_small_train.data",
               "orange_small_train_appetency.labels.txt",
               "orange_small_train_churn.labels.txt",
               "orange_small_train_upselling.labels.txt")

for (i in 1:length(LoadFileList)){
    if (!file.exists(LoadFileList[[i]])){
        if (LoadFileList[[i]]!=DownloadFileList[[i]]) {
            download.file(paste(gitURL,DownloadFileList[[i]],sep=""),
                          destfile=DownloadFileList[[i]])
            gunzip(DownloadFileList[[i]])
        }else{
            download.file(paste(gitURL,DownloadFileList[[i]],sep=""),
                          destfile=DownloadFileList[[i]])}}
}

# Specify strings of na
na_strings <- c('',
  'na', 'n.a', 'n.a.',
  'nan', 'n.a.n', 'n.a.n.',
  'NA', 'N.A', 'N.A.',
  'NaN', 'N.a.N', 'N.a.N.',
  'NAN', 'N.A.N', 'N.A.N.',
  'nil', 'Nil', 'NIL',
  'null', 'Null', 'NULL')

# Load the data
X = as.data.table(read.table('orange_small_train.data',header=TRUE,
                sep='\t', stringsAsFactors=TRUE, na.strings=na_strings))
Y_churn = read.table("orange_small_train_churn.labels.txt", quote="\"")
Y_appetency = read.table("orange_small_train_appetency.labels.txt", quote="\"")
Y_upselling = read.table("orange_small_train_upselling.labels.txt", quote="\"")
```

2. Cleaning the Data

The data we have is challenging and we might consider performing data manipulation for a number of reasons. We list two major ones below. Several `R` libraries can be helpful for easier data manipulations `dplyr`, `tidyr`, `reshape`, `forcats`.

1. Missing values
    - Remove:
        - Most columns contain missing values. Some columns contain only missing values, remove such columns. Some columns contain mostly missing values, remove them as well. There might be columns containing only 1 value other than missing, remove them as well.
    - Impute:
        - For numeric variables, there are a number of ways to deal with missing values. The simplest one is to simply impute the missing value with the mean of observed values. That is, if a particular value is missing in a column just simply substitute `NA` with the mean of the column.
        - For categorical variables, the easiest way to deal with missing values is to treat them as if they are one of the levels. For example, if a categorical feature has levels `A`, `B`, and `C`, then the missing value `NA` can be treated simply as value `D`.
2. Categorical variables with a large number of levels
    - Some categorical features have a large number of different levels. For example, if you try to fit a decision tree using the features, the algorithms will need to make a large number of splits. One way around this is to combine or aggregate levels for which there are a few observations. One suggestions would be as follows: create a new level "low", which combines all existing levels for which we have 1~249 observations;

create a level "medium" that aggregates existing levels for which there are 250~499 observations; create level "high" that aggregates all existing leveles for which there are 500~999 observations; and keep all other levels as they are. There will be a lot of explanatory variables than a simple linear model can handle, and most of those variables do not have any explanatory power.

Coding Hints:

1. How to write a loop over columns of a data.frame (say, X here)?

```
# Method 1
for (i in names(X)){
  CurrentColumn=X[[i]]
  CurrentColumnVariableName=i
  # Then you do the computation on CurrentColumn, using function is.na, and save the result
  cat(i, mean(is.na(CurrentColumn)),'\n')
  # This tells you the percentage of na in the column
}

# Method 2: use dplyr library
# install.packages("dplyr")
library (dplyr)
X %>% summarise_each(funs(mean(is.na(.))))
```

2. How to drop columns of a data.frame indexed by a list?

```
# Method 1
ExcludeVars=c('Var1','Var2','Var100') #for example
idx=!(names(X) %in% ExcludeVars);
XS=X[,!(names(X) %in% ExcludeVars),with=FALSE]

# Method 2: use dplyr library
# install.packages("dplyr")
# library (dplyr)
XS = X %>% select(-ExcludeVars)
```

3. How to convert missing values into factors?

```
# Method 1
i="Var208" #for example

CurrentColumn=XS[[i]]                    #Extraction of column
idx=is.na(CurrentColumn)                 #Locate the NAs
CurrentColumn=as.character(CurrentColumn)#Convert from factor to characters
CurrentColumn[idx]=paste(i,'_NA',sep="") #Add the new NA level strings
CurrentColumn=as.factor(CurrentColumn)   #Convert back to factors
XS[[i]]=CurrentColumn                    #Plug-back to the data.frame

# Method 2: use forcats library
# install.packages("forcats")
library(forcats)
XS[[i]] = fct_explicit_na(XS[[i]], paste(i,'_NA',sep=""))
```

4. How to aggregate a number of factors into new factors?

```
Thres_Low=249;
Thres_Medium=499;
Thres_High=999;
i="Var220" #for example, this one has 4291 levels
CurrentColumn=XS[[i]]                    #Extraction of column
CurrentColumn_Table=table(CurrentColumn) #Tabulate the frequency
```

```
levels(CurrentColumn)[CurrentColumn_Table<=Thres_Low]=paste(i,'_Low',sep="")
levels(CurrentColumn)[CurrentColumn_Table>Thres_Low & CurrentColumn_Table<=Thres_Medium ]=paste(i,'_Medium',sep="")
levels(CurrentColumn)[CurrentColumn_Table>Thres_Medium & CurrentColumn_Table<=Thres_High ]=paste(i,'_High',sep="")

XS[[i]]=CurrentColumn                    #Plug-back to the data.frame
#We got 9 levels after cleaning

# Method 2: use forcats library
library(forcats)
fct_collapse(XS[[i]], Var220_Low = levels(CurrentColumn)[CurrentColumn_Table<=Thres_Low]) # for example

# dplyr library has "recode" and 'recode_factor' functionality as well
```

Follow the procedures/suggestions to clean the data, and provide summary graphs/tables if necessary. You could use your own procedures as well, just state the rationals. Report the columns you removed in a nice way (a figure would be perfect). After you have done cleaning the data, please split them into two parts: train + validation set (~80% of total observations) and test set (~20%).

3. Feature/Variable Selection

After cleaning of data, there should still be quite several variables remaining (hopefully), as the purpose is to present the challenge of machine learning modeling in the presence of large p (high dimensional problem). A proportion of these remaining variables are not important at all and you may want to exclude them in the model (they may not have any explanatory power).
Pick one response variable Y (`churn`, `appetency`, or `upselling`) here, and stick to it for the rest of the homework. Here are some ideas for you to try, for the Y you picked, using the training + validation set only:

- Fit a classifier for Y using only single explanatory variables (no interaction), rank the variables by performance and exclude the worst ones.
  - This might seems intuitive and fast, it does ignore the effect of interactions between variables.
- Fit a classifier for Y using pairs of explanatory variables, and exclude variables that are not among the top performing pairs.
  - Caution: Although it takes care of pair-wise interaction, this takes a long time to compute (p^2 versus p), you may want do the experiment on a smaller set of observations (≤ 1000) first, or do the experiment on a smaller set of variables.
- Fit a random forest model for Y and keep the important variables.
  - Caution: Again, start with a smaller data set when running the experiment.

You only need to do one of them, or come up with your own idea of variable selection. Summarize what variables you keep.

4. Training the Models

Now you have a relatively clean data with a subset of variables, fit a final model using the train + validation set by random forest or boosting. Remember to use cross-validation or out-of-bag error estimate to select tuning parameters. Report the performance of your model in train + validation set.

5. Final Evaluation

Use your final model to make predictions on the test set, report the performance.