15.071: The Analytics Edge **Spring 2020**
Homework 6: Text Analytics and Collaborative Filtering

Out: April 3; Due: April 17, 5 pm.
Please post the assignment in pdf format with file name "Lastname_15071_HW6.pdf".
For each question, please include the main `R` *commands that you used in your submission.*

Problem 1: Predicting Airbnb review scores using text analytics (50 points)
Airbnb is a platform that allows users to list their homes for short-term lease or rental (called a *listing*), and allows other users to reserve lodging. Users who reserve lodging are able to leave a review on the Airbnb website and rate their stay after its completion. The dataset **`airbnb-small.csv`** contains a subset of reviews written for listings in New York City between March 2011 and March 2018, as well as the associated ratings. The variables in the dataset are described as follows:

- **`listing_id`**: an integer key associated with the listing
- **`id`**: an integer key associated with the review
- **`date`**: the date of the review in the format YYYY-MM-DD
- **`reviewer_id`**: an integer key associated with the reviewer
- **`reviewer`**: the first name of the reviewer
- **`comments`**: the text of the review
- **`review_scores_rating`**: the score that the reviewer gave the listing, with 20 corresponding to one star, and 100 corresponding to five stars

The goal of this problem is for you to analyze the text of reviews and to build a model to assess whether any given review is a *positive* review (defined to be four or five stars) or a *negative* review (defined to be one, two, or three stars). You will use a regular "bag of words" data representation of the reviews as has been described in class. Recall that using this representation, for each review we count the number of occurrences of the words in the review. The words (also called terms) are taken as the independent variables, and the data for each review is the number of each word/term in that review. The independent variables will be used to predict whether or not a given review is positive or negative.

When you read **`airbnb-small.csv`** into R, you will want to set the **`stringsAsFactors`** flag to **`FALSE`**. This will prevent the text of the reviews from being turned into factor variables. You can set the flag using the following R code:

```
reviews = read.csv("airbnb-small.csv", stringsAsFactors = FALSE)
```

a) **Preliminary Insights.**

i) For each of the five review scores (20, 40, 60, 80, 100), how many reviews in the dataset have this review score? [**2 pts**]

ii) Create a new column in the dataset that contains the character length of each review. If you saved your dataset as **`reviews`**, you can compute the review lengths using the following command:

```
nchar(reviews$comments)
```

What is the average ("mean") review length for each of the five review scores? (Note: you can use the **`aggregate()`** function.) What do you observe? [**3 pts**] *b)* **Corpus.** Create a corpus (which is a collection of "documents") based on the `comments` column of the dataset. Clean the corpus by applying the following transformations:

i) Convert the text to lowercase.

ii) Remove punctuation.

iii) Remove all stop words.

iv) Remove the word "airbnb".

v) Stem the document.

Look at the first document in the corpus (this corresponds to the first review) after performing the above transformations. What is the text of the document? [**5 pts**]

c) **Sparsifying the Corpus.** Your corpus at this point should still contain words that are very specific (e.g., "turquoise", "b43", or hosts' names) and not helpful due to this specificity. Let us restrict our attention only to words that occur more frequently.

i) Calculate the word frequencies in the corpus. Which are the words that occur at least 900 times? [**2 pts**]

ii) Remove words/terms that occur in less than 1% of the reviews. How many terms do you still have after removing the rarely used terms? [**3 pts**]

d) **Training and Test Split.** Convert your sparsified corpus to a dataframe. This dataframe should contain a row for each document (review), and a column for each word (term) in the sparsified corpus. However, we do not yet have the dependent variable in the dataframe, so here is how to create it. Recall that the dependent variable represents whether the review is positive (four or five stars) or negative (one, two, or three stars). You can create the dependent variable values for each document by running the following R code:

```
documentterms$positive_review = reviews$review_scores_rating >= 80
```

assuming that you have called your document-terms dataframe `documentterms`, and that you have called your reviews dataframe `reviews`.

Next, split the dataframe into a test set and training set. Perform the split so that the training set is comprised of all of the reviews from before December 31, 2017, and your test set is comprised of all of the reviews from January 1, 2018 onwards. [**5 pts**]

e) **Prediction.**

i) Construct a CART model to predict whether a review is positive or negative based only on the text-based features in the dataframe, use the default parameters for `cp` and `minbucket`. (*Note*: Because there are so many terms in this dataset, this may take about 30 seconds on your computer.) Attach an image of the tree. Does the tree agree with what your intuition? What is your interpretation of the terms that the model has selected? [**15 pts**]

By the way, if you want to look for comments containing a particular word in the reviews dataset, use the following R code:

```
reviews[grepl("word", reviews$comments), "comments"]
```

ii) Compute your CART model's accuracy, True Positive Rate (TPR), and False Positive Rate (FPR). How does the accuracy compare to that of a simple baseline model where all reviews are classified as positive? [**10 pts**] *iii*) The "bag of words" representation of text works very well in very many contexts despite its being simplistic in concept. Can you think of examples where the "bag of words" representation might fail? What might you do to improve upon "bag of words"? [**5 pts**]

Problem 2: Recommending Songs to Music Listeners [50 pts]

You will provide recommendations to music listeners on an online platform. We leverage a database that aggregates the ratings of songs on the platform. The challenge is that each user only rates a few songs. The objective is thus to infer each user's ratings of *all* songs in order to develop recommendations.

You have access to the following datasets:

– `Songs.csv`: each observation includes the song's ID, name, year, artist, and genre.

– `Users.csv`: each observation characterizes a user by his/her ID.

– `MusicRatings.csv`: each observation contains the user ID, the song ID, and the corresponding rating.

Start by importing these datasets, using the following commands:

```
songs <- read.csv("Songs.csv")
users <- read.csv("Users.csv")
music <- read.csv("MusicRatings.csv")
```

In addition, you will find a file named **`functionsCF.R`**, which contains two functions:

- **`cf.training.set(rater, item, prop)`** splits the data into a training set and a test set. The function takes as inputs the list of raters, the list of rated items, and a proportion of observations that will go into the training set. It returns the list of rows that will go into the training set. As opposed to our traditional procedure, this function ensures that all raters and all songs appear in the training set.

- **`cf.evaluate.ranks(dat, ranks, prop.validate=0.05)`** finds the "best" number of archetypal users. Ideally, we would use cross-validation, but this is highly time-consuming. Instead, the function further splits the training set into a smaller training set and a validation set. For each number of archetypal users (called "rank"), the function fits a model on the smaller training set and evaluates out-of-sample performance on the validation set. Specifically, it takes as input a dataset (the "full" training set), a list of ranks to try (e.g., **`1:10`** for $0, 1, 2, 3, \cdots, 10$), and a proportion of the training set that will be go into the validation set (set to 5% by default). It returns the R^2, the MAE and the $RMSE$ on the validation set, for each rank. It can then be used to select the rank—without using the test set.

Apply the **`cf.training.set`** function to split the dataset into a training set and a test set, and create an incomplete training matrix, using the following commands:

```
source("functionsCF.R")
set.seed(144)
training.rows <- cf.training.set(music$userID, music$songID, prop=0.92)
music.train <- music[training.rows,]
music.test <- music[-training.rows,]
mat.train <- Incomplete(music.train[,1], music.train[,2], music.train[,3])
```

Finally, we will work with centered matrices in this problem. This makes use of the `biScale` function. We set **`row.scale`** and **`col.scale`** to **`FALSE`** to keep the standard deviation of the data the same.

```
set.seed(15071)
mat.scaled <- biScale(mat.train, maxit=1000, row.scale = FALSE, col.scale = FALSE)
```

a) Call **`set.seed(123)`**. Then use the **`cf.evaluate.ranks`** function with **`prop.validate=0.05`**. Plot the model's estimated performance, as a function the number of archetypal users. How many archetypal users do you choose? Is it what you would have intuitively guessed? Please comment briefly. **[15 pts]**

b) Call **`set.seed(15071)`**. Then with the number of archetypes you have selected, fit a collaborative filtering model (called **`fit`**) with the training set using the **`softImpute`** function. Use the model to make test-set predictions. What is the out-of-sample R^2 of the model? **[10 pts]**

c) For the next two questions we consider user Daisy (**`userID`** = 1584). Using the model what is Daisy's predicted rating of song 131? Song 156? **[10 pts]**

[**Hint:** Use the `impute` function.]

d) Daisy (`userID` = 1584) has just selected the *Surprise me!* option, which automatically builds a playlist for her. Which five *new* songs would you put at the top of Daisy's playlist (the playlist should not include songs that Daisy has already rated)? Report each song's ID, name, artist, and genre. Compare them to the five songs that Daisy has rated the highest. Comment briefly. **[15 pts]**

[**Hint:** May want to use the `top_n` function.]