15.071: The Analytics Edge
Spring 2019

Homework Assignment #6

Due at 5pm on April 12, 2019

You must post your completed assignment on Canvas.

Note: this homework does not have an online portion.

Problem 1: Predicting skin moisturizer sales using time series (50 points)

Eczema (or dermatitis) is a group of diseases that cause the human skin to become inflamed and irritated.[1] In this exercise you will construct models to predict weekly sales volume for a skin moisturizer that reduces the irritation caused by eczema. You will use the dataset `WeeklyMoisturizerSales.csv`, which contains weekly sales volume data for a particular eczema skin moisturizer, for every week between August, 2010 and July, 2015. The variables in the dataset are described as follows:

- **`Date`**: Dates for the beginning of each week in the format YYYY-MM-DD.
- **`MoisturizerSales`**: An approximation of skin moisturizer sales volume for each week. (Note: the values are scaled so that they are between 0 and 100.)
- **`GoogleTrendVolumeEczema`**: An approximation of the number of Google searches that contain the word "eczema" for each week. (Note: the values are scaled so that they are between 0 and 100.)
- **`Month`**: The observation month (corresponding to the beginning of each week) given as the name of the month.

The goal of this problem to construct and compare some different time series models to predict future weekly sales, namely:

- an autoregressive model,
- an autoregressive model with month factors, and

[1]See the article https://en.wikipedia.org/wiki/Dermatitis for more information

- an autoregressive model with month factors as well as the previous week's Google searches of "eczema".

1. **Preliminary insights (5 points).**
   (a) Plot the variable `MoisturizerSales` versus time. What patterns do you observe?
   (b) Plot the variable `GoogleTrendVolumeEczema` versus time. What patterns to you observe? How does this plot compare with the previous plot?

2. **Creating lag variables, and training and test set split (5 points).**
   - **Creating lag variables.** Modify the dataset `WeeklyMoisturizerSales.csv` by adding two new variables – one lag variable with the data for the previous week's moisturizer sales, and another lag variable for the previous week's Google searches that contain "eczema".
   - **Training/test split.** If you saved your dataset as `MoisturizerData`, use the following R code to construct a training set that contains observations with `Date` variable values before and including 2013-06-30:

     ```
     training = subset(MoisturizerData, as.Date(Date) <= as.Date("2013-06-30"))
     ```

     Use the following R code to construct a test set that contains the observations with the other (later) `Date` variable values:

     ```
     testing = subset(MoisturizerData, as.Date(Date) > as.Date("2013-06-30"))
     ```

3. **Construct and compare three models (40 points).**
   (a) (10 points) Construct a simple autoregressive time series model to predict weekly moisturizer sales, by using linear regression. Your dependent variable should be `MoisturizerSales`, and your independent variable should be the lag variable for the previous week's moisturizer sales. Compute and report the R^2 and the $RMSE$ for this model.
   (b) (10 points) Now construct an autoregressive time series model that additionally includes the factor variable `Month`. What is the R^2 and $RMSE$ for this model? What is your intuition for why one model performs better than the other?
   (c) (10 points) Thirdly, construct an autoregressive model that includes the factor variable `Month` and the additional lag variable for the previous week's Google searches of "eczema". Compute and report the R^2 and $RMSE$ for this model. Once again, what is your intuition for this model's performance?
   (d) (5 points) For your last model, compute the OSR^2 and out of sample $RMSE$. What is your overall assessment of the quality of this model?
   (e) (5 points) What other data/variables might you try to include in a model to predict weekly moisturizer sales?

Problem 2: 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*), as well as to allow 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 = F)
```

1. **Preliminary Insights.**
   (a) (2 points) For each of the five review scores (20, 40, 60, 80, 100), how many reviews in the dataset have this review score?
   (b) (3 points) 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?

2. **Corpus.** (5 points) 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:

   (a) Convert the text to lowercase.

   (b) Remove all stop words.

   (c) Remove the word "airbnb".

   (d) 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?

3. **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 do not occur so rarely.

   (a) (2 points) Calculate the word frequencies in the corpus. Which are the words that occur at least 1,500 times?

   (b) (3 points) 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?

4. **Training and Test Split.** (5 points) 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 (inclusive), and your test set is comprised of all of the reviews from December 31, 2017 onwards (non-inclusive).

5. **Prediction.**

   (a) (3 points) Create a simple baseline model. Describe the baseline model you have created. What is your baseline model's accuracy?

   (b) (15 points) Construct a CART model to predict whether a review is positive or negative based only on the text-based features in the dataframe. (*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?

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"]
```

(c) (7 points) Compute your CART model's accuracy, True Positive Rate (TPR), and False Positive Rate (FPR). How does the accuracy compare to that of your baseline model?

(d) (5 points) 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"?