CptS 475/575: Data Science, Fall 2019

Assignment 5: Classification

Release Date: October 18, 2019 **Due Date:** October 28, 2019 (11:59 pm)

General instruction: The first part of this assignment will assess your understanding of logistic regression. The second part of this assignment will require you to take a set of articles from a real world newspaper and classify them based on which section they belong to. The problem is broken into four sections with points assigned to each.

Your solution will be submitted as a PDF file, which must **include your full, functional code and relevant results as stated in each part**. You are encouraged to use R Markdown to prepare your file.

1. (20%) Suppose we collect data for a group of students in a statistics class with variables:
X_1 = hours studied,
X_2 = undergrad GPA,
X_3 = PSQI score (a sleep quality index), and
Y = receive an A.
We fit a logistic regression and produce estimated coefficient, $\beta_0 = -7$, $\beta_1 = 0.1$, $\beta_2 = 1$, $\beta_3 = -.04$.

a. Estimate the probability that a student who studies for 32 h, has a PSQI score of 12 and has an undergrad GPA of 3.0 gets an A in the class. Show your work.

b. How many hours would the student in part (a) need to study to have a 50 % chance of getting an A in the class? Show your work.

c. How many hours would a student with a 3.0 GPA and a PSQI score of 3 need to study to have a 50 % chance of getting an A in the class? Show your work.

2. For this question, you will a naïve Bayes model to classify newspaper articles by their section.

a. Data collection (5%)

First you will need to gather your data from the web API for your newspaper of choice. Because most web APIs return JSON objects, you may want to use the fromJSON function in the RJSONIO package to make your API requests. Alternately, you can automate some of the query generation using the GuardianR package. I suggest using the Guardian, which has fairly lenient query limits, and makes getting an API key very simple (https://open-platform.theguardian.com/access/), but you may use any newspaper you like, so long as it includes section headings with its articles and has at least six sections including **world news**, **science**, and **economy/business**. You may choose the three other categories as you see fit. You may also want to choose a paper that lets you download full article text, as some papers like the New York Times only give you the lead paragraph. To get full credit for this portion you must include your full code and demonstrate (by showing **summaries of your data**) that you have collected a minimum of **approximately 6,000 articles** from a newspaper in six sections, and that approximately **1,000 articles are in each section**.

b. Data cleaning (5%)

The data you have now collected is likely very dirty. Examples of issues you are likely to run across are invalid characters, HTML tags, links and other non-article text, varied cases and punctuation. All of these will cause problems for the tokenization step that comes next. For this portion, you will clean up the article bodies you collected by removing the above mentioned messiness, and any other non-uniform elements you come across in your data. When you are finished, each article body should contain only plain, lower case text and no punctuation. To demonstrate that you have completed this task, include **your code**, and print out the **cleaned content of a single article (randomly chosen from the collection)**.

If you are unable to complete these first two portions of the assignment, a pre-collected and cleaned dataset is available for you on the course webpage, so that you may complete the remainder of the assignment.

c. Tokenization (25%)

In order to use Naïve Bayes effectively, you will need to split your text into tokens. It is common practice when doing this to reduce your words to their stems so that conjugations produce less noise in your data. For example, the words "speak", "spoke", and "speaking" are all likely to denote a similar context, and so a stemmed tokenization will merge all of them into a single stem. R has several libraries for tokenization, stemming and text mining. Some you may want to use as a starting point are tokenizers, SnowballC, tm respectively, or alternatively quanteda, which will handle the aforementioned along with building your model in the next step. You will need to produce a term-document matrix from your stemmed tokenized data. This will have a very wide feature set (to be reduced in the following step) where each word stem is a feature, and each article has a list of values representing the number of occurrences of each stem in its body. Before representing the feature set in a non-compact storage format (such as a plain matrix), you will want to remove any word which appears in too few documents (typically fewer than 1% of documents, but you can be more or less stringent as you see fit). You may also use a boolean for word presence/absence if you find it more effective. To demonstrate your completion of this part, you can simply print the nonzero portion of the **feature vector for the article from part 2**. (If you did not do part 2, just give the raw version of the article body as well.)

d. Classification (45%)

For the final portion of this assignment, you will build and test a Naïve Bayes classifier with your data. First, you will need to use feature selection to reduce your feature set. A popular library for this is caret. It has many functionalities for reducing feature sets, including removing highly correlated features. You may wish to try several different methods to see which produces the best results for the following steps.

Next, you will split your data into a training set and a test set. Your training set should comprise approximately 80% of your articles, however, you may try several sizes to find which produces the best results. Whatever way you split your training and test sets, however, you should try to ensure that your six article categories are equally represented in both sets.

Next, you will build your Naïve Bayes classifier from your training data. The e1071 package is most commonly used for this. Finally, you can use your model to predict the categories of your test data.

Once you have produced a model that generates the best predictions you can get, **print a confusion matrix** of the results to demonstrate your completion of this task. For each class, give scores for **precision** (TruePositives / TruePositives+FalsePositives) and **recall** (TruePositives / TruePositives+FalseNegatives). To do this, you may want to use the confusionMatrix() function.