IEOR 242: Applications in Data Analysis, Fall 2021

Homework Assignment #5

November 13, 2021

Problem 1: Predicting Useful Questions on Stack Exchange (100 points)

Stack Overflow is a very popular question-and-answer website, featuring questions on many different computer programming topics. The Stack Exchange Network also features many different spin-off sites on a wide variety of topics. Users of Stack Overflow and its subsidiary Stack Exchange communities can post questions and also provide answers to questions. Users who have earned certain privileges can also vote on the quality/usefulness/helpfulness of both questions and answers. For each question (or answer), privileged users who have read the question can elect to provide either an "upvote" or a "downvote" for the question. An upvote constitutes an endorsement of the question's usefulness for other users, whereas a downvote signals that the question is likely to be especially not useful (or has been sloppily written, or does not show enough research effort, etc.). The "score" of a question is equal to the total number of upvotes for that question minus the total number of downvotes.

In this problem, you will work with question data from Stack Exchange, made available as part of the Stack Exchange Network "data dump".[1] The particular dataset that you will work with contains all of the questions asked during 2020 in the Cross Validated[2] Stack Exchange community, which concerns topics such as statistics, machine learning, data analysis, and data visualization. The data is contained in the files `stack_stats_2020_train.csv` and `stack_stats_2020_test.csv` and has been randomly split, with 70% in the training set and 30% in the testing set. Each observation records text data associated with the title and the body of the associated question, text data for the tags of the question, and also the score of the question. Table 1 describes the dataset in further detail.

After a question is posted, it would be beneficial for Stack Exchange to get an immediate sense of whether the question is useful or not. With such knowledge, the website could automatically promote questions that are believed to be useful to the top of the page. With this goal in mind, in this problem you will build models to predict whether or not a question is useful, based on its tags and the title and body text data associated with each question. To be concrete, let us say that **a question is useful if its score is greater than or equal to one.**

As compared to previous homework exercises in this course, this exercise is purposefully open

[1] https://archive.org/details/stackexchange
[2] https://stats.stackexchange.com/

ended. Therefore, your submission should be in the form of a brief report, with one section dedicated to each of the parts (a), (b), and (c) outlined below.

Table 1: Description of the dataset `stack_stats_2020`.

Variable	**Description**
`Title`	Title text of the question
`Body`	Main body text of the question (in html format)
`Tags`	List of tags associated with the question (the format is `<tag1><tag2>...`)
`Score`	Score of the question, equal to total number of upvotes minus total number of downvotes

a) (30 points) Start by cleaning up the dataset to get it into a form where we can apply statistical learning methods to predict our dependent variable of interest. Please briefly describe each step that you took to clean and process the data. Here are some suggestions:

i) The Python library `BeautifulSoup` is useful for dealing with html text. In order to use this library, you will need to install it first by running the following command:

```
conda install beautifulsoup4
```

in the terminal. In the code, you can import it by running the following line:

```
from bs4 import BeautifulSoup
```

ii) When converting html texts that include multiple lines of texts to plain texts using `get_text()` from `BeautifulSoup`, the output texts still contain one type of html string, `\n` or the linebreak character. In other words, `get_text()` does not remove the linebreak characters. You will need to remove them using a proper method of your choice.

iii) You should also inspect several texts from `Body, Tags` and `Title` to see if there is any other transformation needed. You may need to write a custom transformation function.

iv) Once you have all texts in plain texts, you can use the Python library `nltk` to do text cleaning and generate document term matrices as we did in Lab.

v) Notice that there are words that appear in more than one column. For instance, a question might have the term 'regression' in its title, body and tags. However, the terms 'regression' that appear at the three positions of this post may give different implications about this post. (For example, using 'regression' in the title may signal something different than 'regression' as a tag.) For this reason, the three types of text data that we are dealing with should be processed *independently*, and you should use different column names (e.g., `regression_title`, `regression_body`, `regression_tags`).

b) (40 points) Use your analytics skills to build the best model that you can for predicting useful questions. You are free to try whichever approaches you like, but you should try at least **four** distinct methods that have been discussed in class (e.g., Logistic Regression, LDA, CART, Random Forests, Boosting) and you should evaluate no more than a handful (i.e., 4-10) of candidate models on the test set. For Random Forests and Boosting, it is OK to not cross validate all of the parameters if it takes too much time on your computer – instead you may use some reasonable values of these parameters and/or only cross validate some of them.

Report on the details of your training procedures and final comparisons on the test set. Use your best judgment to choose a final model and explain your choice. Use the **bootstrap** to asses the performance of your single chosen final model in a way that properly reports on the variability of the relevant performance metrics (accuracy, TPR, and FPR).

c) (30 points) Now, let us consider how Stack Exchange might use your model. In particular, consider the following scenario. When a user navigates to the Cross Validated page, they are automatically presented with the 15 most recently submitted questions, in order of most recent first (this is not necessarily true in reality, but let's pretend that it is). Suppose that Stack Exchange would like to keep these same 15 most recently submitted questions on this page, but they would like to rearrange the order of the questions so as to show the most useful questions at the top of the page. Suppose further that Stack Exchange believes that most users are extremely impatient and will only pay attention to the single question at the very top of the page, and therefore they would like to **maximize the probability that the top question is useful**.

i) Think about how to select a model, among many different models, to best accomplish the goal of maximizing the probability that the top question is useful. Comment on the precise criteria that you would use (e.g., "I would select a model with the highest accuracy" or "I would select a model with the highest TPR", etc.) and note that your answer may involve multiple performance metrics if you wish. Explain your response.

ii) Revisiting the models that you have built in part (*b*), can you identify a specific model that best accomplishes the goal? (Note that this model may be different from the final model you selected in part (*b*), and you are allowed to do some re-training in addition to re-evaluating your models.) How much does the model you selected improve upon Stack Exchange's current approach of showing the most recent posts first (described above)? In particular, use the results of your model on the test set to give a precise numerical estimate of the increase in the probability that the top question is useful. If you like, you may use a back-of-the-envelope "on average" style of analysis as part of your reasoning to answer this question.