Assignment 3

Logistics

You must create an IPython notebook for the problem. Please name the notebook *"HW3_YourName"*. Upload the notebook to Canvas.

The assignments are due on **August 3rd, before the beginning of class**.

Deliverables. The notebook should be split by sections, one for each question. Each section should contain the following parts:

1. A heading for each question (use a Markdown cell).
2. For each question, write a short description of your approach (another Markdown cell).
3. Write the code, and show the results or plots.

Introduction

Suppose we have a bunch of URLs and we want to know their adult-rating (i.e., is the url P, or G, or X, or R). This task is difficult for computers, but easy for humans, and this has led to the growth of crowdsourcing: get a bunch of humans to give ratings to urls, but use automated techniques to figure out how much to trust each person's ratings.

We are going to use the data from a paper by Ipeirotis et al. available here. This details an experiment run on Amazon's `Mechanical Turk` crowdsourcing system. They ask a bunch of raters (called "turks") to rate several urls, but they already know the answers (the true categories) for a few urls, called the "gold set". The ratings of the turks on the gold set thus allows us to judge their accuracy.

[Q1 10 points] Read in data

Read in the files `gold.txt` and `labels.txt`. The `gold` DataFrame should have columns `url` and `category`, while the `labels` DataFrame should have columns `turk`, `url` and `category`. You will have to pick the right separator.

[Q2 10 points] Split into two DataFrames

Split the `labels` DataFrame into two: `labels_on_gold` and `labels_unknown`, the former containing all rows where the url is present in the gold set, and the latter one contains all remaining rows of `labels`.

[Q3 10 points] Compute accuracies of turks

Create a `rater_goodness` DataFrame that is indexed by turk, and has two columns: the number of ratings, and the average correctness of ratings for each turk (both on gold set urls).

[Q4 10 points] Odds ratios

If someone is correct p fraction of the time, the *odds* of success are defined as:

$$\text{odds} = \frac{p}{1-p}.$$

Attach a column called `odds` to the `rater_goodness` DataFrame, using the average correctness of the turk as his or her p.

[Q5 10 points] Most accurate turks

List the top 10 most accurate turks who have rated at least 20 gold set URLs.

[Q6 10 points] Rating counts versus accuracy

One may imagine that a committed and accurate turk will rate lots of URLs. On the other hand, perhaps it is only the spammers who have the time to rate lots of URLs.

Is number of ratings by a turker on gold set URLs related to his or her accuracy? There's no fixed answer; just try to show some evidence for your answer.

[Q7 13 points] Overall predicted odds

Consider each url u that is **not** in the gold set, and each category c. For the pair (u, c), calculate the product of odds of all turks who (a) rated url u as category c, and (b) have rated more gold set urls than 75% of all turks.

We shall call such products of odds the *overall odds* henceforth.

[Q8 13 points] Predicted categories

Create a DataFrame (called `result_75`, whose index is URLs not in the gold set, and with two columns called `top category` and `top odds`. The `top category` should be the category with the highest overall odds for that url, and the `top odds` should be the overall odds for that `top category`.

These are our predictions, and the confidence we have in them (higher overall odds implies greater confidence). If you want, you can check to see if the predicted categories make sense.

[Q9 14 points] Predicted categories using more turks

Questions 7 and 8 above only considered the ratings of turks who had rated enough gold set URLs, so we were relatively more confident about their accuracies. What happens if we loosen this restriction?

Repeat the code of Q7 and Q8, but replacing 75% by 25% in the description of Q7 (i.e., we also consider turks who have far fewer gold set ratings). Call this `result_25`.

Now let's see how these two results compare. Create a DataFrame where both the index and the columns are the various categories, and the cells contain the number of urls with these as the top categories according to `result_75` and `result_25`.

For example, the cell corresponding to the row `category=R` and the column `category=G` would be the number of URLs that were predicted to be **R** by `result_75` but predicted to be **G** by `result_25`.

Where are the most errors?

Hint: You might want to use the `pd.crosstab` function.