

## CS 5100 Foundations of AI

### Project: Identifying Personal Attacks in Wikipedia Comments

Value: 20%

Note that there are no slip days or extensions possible for the project.

In this project, given a publicly available dataset, you will work on a classification task.

You will be using the “Wikipedia Talk Labels: Personal Attacks” data set, Version 6 of 22 Feb 2017, available from [https://figshare.com/articles/Wikipedia\\_Talk\\_Labels\\_Personal\\_Attacks/4054689](https://figshare.com/articles/Wikipedia_Talk_Labels_Personal_Attacks/4054689) .

I am grateful to the Wikimedia Foundation and [Ellery Wulczyn, Nithum Thain and Lucas Dixon](#) who worked on this dataset and related materials, for making this publicly available.

Here is a description of the data set from that page: “This data set includes over 100k labeled discussion comments from English Wikipedia. Each comment was labeled by multiple annotators via Crowdfunder on whether it contains a personal attack. We also include some demographic data for each crowd-worker. See our [wiki](#) for documentation of the schema of each file and our [research paper](#) for documentation on the data collection and modeling methodology. For a quick demo of how to use the data for model building and analysis, check out this [iPython notebook](#).” [now Jupyter notebook]. Please note that this data set uses real data from Wikipedia comments. As such, these comments include some adult language.

You will develop a machine learning model to classify discussion comments on whether they contain a personal attack or not. You will start learning about this project using the ‘strawman’ program -- the Jupyter/iPython notebook referred to above -- that is provided in Piazza. See Appendix A of this assignment for an introduction to installing Jupyter and using Jupyter notebooks. You will use a variety of mechanisms to improve upon the performance (accuracy) of the system. *The focus will be on understanding what you are doing and why.*

The next sections describe what is expected from you.

### Section 1. Get the tools ready

1. First make sure you have Python installed. You can use the latest versions of either Python 3.x or Python 2.x. If you have not installed Python before, it is better to use Python 3. Note that there are differences between Python 2 and Python 3, like the fact that *print* needs parentheses in Python 3. Also note that the *urllib* module in Python 2 has been split into parts and renamed in Python 3 to *urllib.request*, *urllib.parse*, and *urllib.error*.
2. Make sure you have Jupyter also installed, and you know how to use it. One of the outputs for the project will be a Jupyter notebook. See Appendix A for help on installing Jupyter and creating and using Jupyter notebooks. Alternatively, you can also use any IDE to develop Python programs, and you can import them from within Jupyter notebooks. But Jupyter makes it easy to develop Python code incrementally, and to document what you’re doing as you go.
3. The Jupyter (iPython) notebook – the strawman code – is available from the Resources page of our course Piazza site. Download the Jupyter notebook **WikipediaTalkDataGettingStarted.ipynb** from the Resources/Project section of Piazza and save it on your laptop. This notebook has two parts: we

will focus on the first part, titled “Building a classifier for personal attacks.” Just because we have this code does not mean we’re done with the project – we’ve just started! *You will work on multiple variations of this code, focused on improving performance and accuracy.*

## Section 2. Run the strawman

4. Read the code and try and understand it. There may be portions you have to look up and understand as you go. That is ok – there’s lots of new material here.
5. Run the notebook, especially the part titled “Building a classifier for personal attacks”. You may have to fix the code if you are using Python 2 (remove `.request` after `urllib` in cell 2, in the `download_file` method definition.) You may also want to experiment with tweaking some of the parameters to see how they change the results.
6. Take a good look at the metric AUC ROC (Area Under the Receiver Operating Characteristic Curve) – this is a measure of how good the machine learned model is. We will be working to improve this measure by improving every aspect of developing the model: data cleaning, feature extraction, modeling etc. A portion of the points for this assignment will depend on the improvement you are able to show.
7. The data is in two files. The code downloads it from a URL every time. You may want to download it once, and then load the local copy each time afterwards. For convenience, these files are also in the Project section in Piazza. File **`attack_annotated_comments.tsv`** has the text of the comments, along with a few other parameters. File **`attack_annotations.tsv`** has ratings on each comment from 10 annotators on different kinds of attacks. We are interested only in the ratings in the final ‘`attack`’ column. `Rev_id` is the column that links the data in the two files. You will need to link the data from the first file to the `attack` ratings in the second file. You will also need to decide how to deal with ten people’s ratings for each comment.

Take a sample (say 100 rows) of data from each file. Examine the data to get a feel for it.

## Section 3. Create the Best “Personal Attacks Classifier”

The following steps may be done in any order you prefer, though step #8 is best done first, and step #12 is done after step #11. Keep track of what you do, as you will need to write about your trials/experiments.

8. **Text cleanup:** Often, the biggest chunk of time and effort spent in machine learning and big data projects is to get the data cleaned up and ready to be used. The strawman code does some simple cleaning of data, replacing newlines and tabs with spaces. See if you can come up with other, better ways you can clean up the data to improve performance and accuracy.
9. **Metrics:** In addition to the `roc_auc_score` in the strawman code, you must add metrics for precision & recall, and a confusion matrix (see `sklearn.metrics.precision_recall_fscore_support` and `sklearn.metrics.confusion_matrix`).

Note that the rows in the `attack_annotated_comments.tsv` file have a column indicating if they are in the training, test or validation (dev) set. Consider mingling the training and test sets and using n-fold cross-validation where possible. What value of n do you think will be good? Why?

10. **Feature Extraction:** This is where you decide which features you will use to classify the comments and related data. Start with using a bag of words representation using words from the comments by themselves (word unigrams). Try word and character n-grams, individually, and together, in various combinations. The strawman code uses two feature extraction methods (*CountVectorizer* and *TfidfTransformer*). You could use *TfidfVectorizer* as an alternative to the combination of these two, to generate features. You may need to use *FeatureUnion* as well.

See if you can add other meaningful non-word features. For example, is 'year' useful as a feature? Why or why not?

11. **Modeling the data:** Try at least 3 modeling methods (other than the *LogisticRegression* module used in the strawman code), to see if any of them helps improve system accuracy. Here are some methods you could try: *MultinomialNB*, *RandomForestClassifier*, *LinearSVC* (Linear Support Vector Machines) and Multilayer Perceptrons (*MLPClassifier*). Feel free to try other methods too.

Experiment with using different ML algorithms with the goal to understand why some perform better. More is good. But stop and think about the results you get from each algorithm.

Finally, pick the model that improves metrics overall for this problem.

12. **Tuning hyperparameters:** Once you find a classifier model that gives you great results, try optimizing the classifier's parameters. For example, assume that *LinearSVC* gave you the best results with default hyperparameters. *LinearSVC* has hyperparameters like *C*, *class\_weight* and *kernel*. A combination of such parameters may improve performance even more. How would you identify the best combination(s)? Check out hyper-parameter tuning methods such as *GridSearchCV* and *RandomizedSearchCV*. Again, keep track of your work.

Once you are happy with the results you are getting, you can finish up the report and submit.

**Note that there are no slip days or extensions possible for the project.**

## Section 4. Preparing your submission

Your submission will consist of three parts:

1. A Jupyter notebook (filename: **yourFirstName\_yourLastName\_Project.ipynb**) containing **your best version of the classifier to solve this problem**, along with appropriate comments and explanatory text about how you went about developing this classifier, especially about the questions (a) to (i) below. The notebook should show both the code and the results from running the code. Please indicate if you are using Python 2 or Python 3. This notebook should clearly include answers for the following questions:
  - a. What are the text cleaning methods you tried? What are the ones you have included in the final code?
  - b. What are the features you considered using? What features did you use in the final code?
  - c. What optimizations did you add in your code, if any?
  - d. What are the ML methods you tried out, and what were your best results with each method? Which was the best ML method you saw before tuning hyperparameters?

- e. What hyper-parameter tuning did you do, and by how many percentage points did your accuracy go up?
- f. What did you learn from the different metrics? Did you try cross-validation?
- g. What are your best final Result Metrics? By how much is it better than the strawman figure? Which model gave you this performance?
- h. What is the most interesting thing you learned from doing the report?
- i. What was the hardest thing to do?

The notebook and its contents will be evaluated for a total of 15 points, of which 2 points will be based on the best performance (in terms of recall and precision) you were able to achieve.

2. Slide deck: A slide deck, (filename: **yourFirstName\_yourLastName\_ProjectSlides.pptx**) less than or equal to 5 slides, in Microsoft PowerPoint format. The deck must answer questions in items (a) to (g) in item 1 of submissions above. Clarity will be much appreciated. You do not have to detail the problem specification in the deck – let's assume the audience knows that.
3. Presentation of your slide deck. A video of you presenting the slide deck (filename: **ProjectVideo** with an appropriate video file extension). Use your phone or your laptop to capture the video. We don't expect any fancy video, but the video and presentation must be clear and audible. The video must be a minimum of 3 minutes and a maximum of 5 minutes long. On the Project Presentation day, during our regular class discussion session, we will play all the slide decks in turn. Each slide deck and presentation will be evaluated for a maximum of 5 points.  
Since phone recordings may have resolution issues, you may want to check out software for screen recording. Your TAs have found two links to tutorials for Mac/PC screen recording.  
MacBook: <https://support.apple.com/en-us/HT208721>  
Windows10: <https://helpdeskgeek.com/windows-10/how-to-record-your-screen-on-windows-10/>  
These links are just suggestions. Please use whatever hardware/software you find appropriate. Make sure you play it back after recording to ensure it is a decent recording of your presentation. Do not wait till the last minute to record your presentation. Try it out earlier, so you know how it works when you are ready to do the final recording.

## Section 5. Submission

1. When you have all components of your submission ready, please name the files using the format:  
**yourFirstName\_yourLastName\_Project.ipynb** ,  
**yourFirstName\_yourLastName\_ProjectSlides.pptx**, and  
**yourFirstName\_yourLastName\_ProjectVideo** (and the extension for the video file)
2. Zip up the files into one .zip file, again with the **yourFirstName\_yourLastName\_Project** name and submit the zip file via Blackboard.
3. Please also email a copy of your slides to your instructor.

Please do not plagiarize. You will be evaluated on the content (not on style – this is not a writing course), but readability and correct spelling will help you get higher scores.

Hope this will be another great learning experience for you!

## Appendix A: A quick introduction to Installing and Running Jupyter

Installing Jupyter: Check out <https://jupyter.readthedocs.io/en/latest/install.html>

Running Jupyter Notebook: See <https://jupyter.readthedocs.io/en/latest/running.html>

You can also download the open-source Anaconda Distribution from <https://www.anaconda.com/distribution/>. This will download and install a number of relevant Python packages, including Jupyter (iPython), scikit-learn, NumPy, Pandas etc., and help you manage libraries, dependencies etc. using Conda.

Here are some important steps in starting and using Jupyter.

Once you have Jupyter Notebook installed, you can run from your terminal/shell window, using the command:

```
jupyter notebook
```

Jupyter prints out a number of logging steps on the terminal window, and then opens a notebook on your default browser:

```
Serving notebooks from local directory: /Users/chandra
[I 16:06:24.671 NotebookApp] 0 active kernels
[I 16:06:24.671 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/?token=.....
[I 16:06:24.671 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```

...

If for some reason the notebook does not open, you can open a browser and navigate to the URL in your terminal window, or just try:

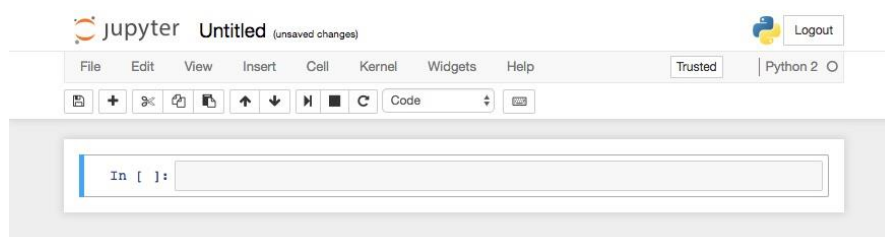
<http://localhost:8888>

To stop the Jupyter Notebook process at any time, press CTRL+C in the terminal window, and hit Y with ENTER to confirm. Jupyter will print out a confirmatory message and exit.

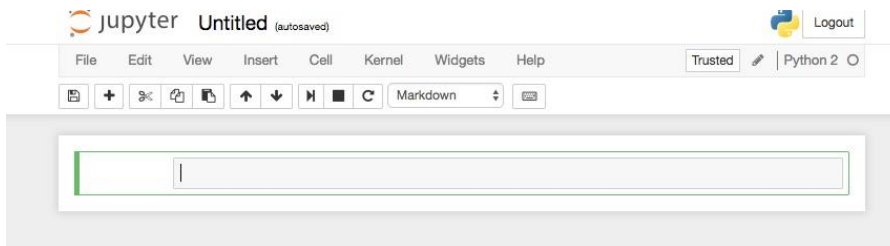
Jupyter Notebook is a powerful tool with many features. Here are a few commands to get you started on using the tool.

Create a new notebook file, using the **New > Python 3** dropdown menu (near the top-right of the window). (You could create a Python 2 notebook, if you prefer, in which case you will use **New>Python 2.**)

This opens a notebook, which looks like this:



You can run Python code in the cell. Or you can type in text, by changing the cell to Markdown mode. You can change the mode by clicking on the **Code**> dropdown, and choosing Markdown, or by the menu choice **Cell>Cell Type> Markdown**



You can now write down notes or instructions or explanations in Markdown. Here is a quick intro to Markdown, if you need it: <https://guides.github.com/features/mastering-markdown/>

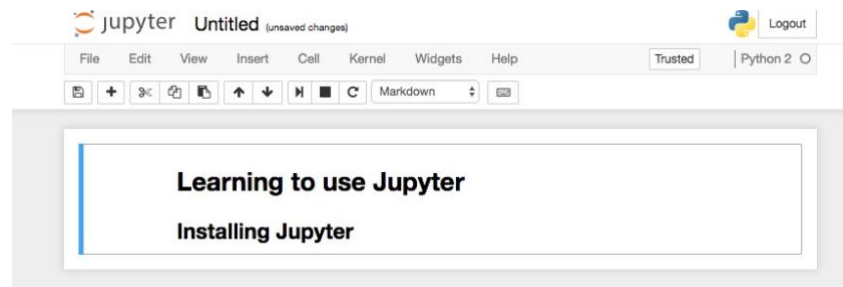
(You can also write equations with  $$$$ s using LaTeX notation, if you are familiar with that.)

Type in some text and hit CTRL-ENTER to turn it into rich text. For example, if you type in something like:

# Learning to use Jupyter

## Installing Jupyter

in the cell and press CTRL-ENTER, you will see something like:

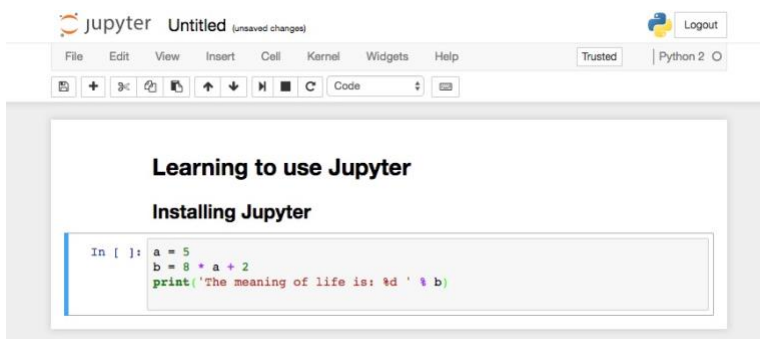


Click on this cell and press Option+ENTER (or ALT-ENTER) to insert a new cell. (Or click on the Run icon below the menu items). Try entering some code in this new cell, say:

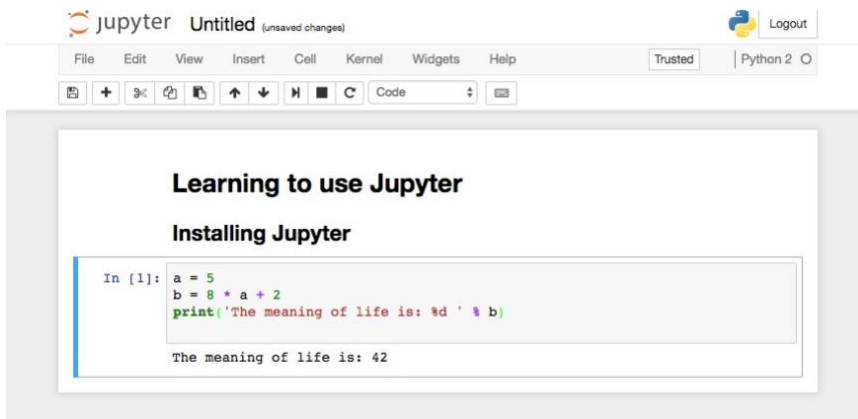
```
a = 5
```

```
b = 8 * a + 2
```

```
print('The meaning of life is: %d ' % b)
```



Run this code, by pressing CTRL+ENTER. You will see something like:




You can save your work, using **File>Download as** and choosing to save it as an iPython notebook (what you normally want), or as a Python program, etc. You can then do **File>Close and Halt** to exit. You will still have to Logout of the opening Jupyter browser screen, and CTRL-C in the terminal/shell window to stop the Jupyter server. Jupyter will output a set of log statements and exit.

Try **Help>User Interface Tour** to understand various UI features.

When you next want to work on the saved screen, use the shell command

```
jupyter notebook
```

from any folder, and navigate to the folder where you had your iPython notebook. Click on your notebook file, and you can now edit it or run it. To run it cell by cell, just keep pressing the run icon . Remember that some cells can take time to run, so patience is your friend.

*Congratulations, you have now entered text and code and executed them in a Jupyter notebook!*

You can now write complex Python code (or use it for other languages), including importing modules, having inline plots (with the magic `%matplotlib inline` command), etc., just like in any Python IDE.