**Eth. AI, EC1, Adversarial Attacks**

**Due Week 12, Thursday, 11:59 PM**

**Goal:** Make the LR model from HW4 more robust via adversarial training.

**Resources:** Tutorials on python and sklearn library can be found both in [W3Schools](https://www.w3schools.com/python/default.asp) as well as [here (Python course Slides](https://ind657-my.sharepoint.com/:f:/g/personal/jrusert_pfw_edu/Ei_EBaJU-IJBgpwM2A7mmvUB-GrO9jUSvSq0CgMgwZgwYw?e=DJcAgX)). Also, sklearn has demonstrations of models and functions on the specific model pages as well (see below links).

**Models:** You will be working with 1 type of model for this homework, 1 from the sklearn python library:

[Logistic Regression](https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html)

Even though this builds off HW4, it still should be handed in separately in its own .py and report file.

## Adversarial Training

**Goal:** In order to strengthen your LR model from HW4 against the untargeted attack, you will be utilizing the same attack to generate additional training data (from your training data). You will use the original training data, plus adversarial training examples to train a new LR model and test against the untargeted text examples from HW4.

**Coding Goals:**

* Create a function, **adversarial\_training**, which creates a new training file consisting of the original training examples and new attacked examples which are created from the original training examples. (**IMPORTANT:** No test data should be used for training or creating new training data).
  + For each original training example, you should create **k** adversarial examples by passing the example to the untargeted attack function from HW4.
  + Write the original example and its label (same format as the training csv) into a new csv: **adversarial\_train\_rotten-tomatoes.csv**
  + Write all **k** adversarial examples into the same output file.
* Create a second function which trains a new LR on the adversarial training data. This should simply create a new LR and new vectorizer as you’ve done for the original training data in HW4.
* Create a third function which tests both the original LR/vectorizer on the original test and adversarial test data, as well as tests the adversarial trained LR/vectorizer on the original test data and adversarial test data.
* Produce results for the following:
  + Original LR/vectorizer scores for the original test data and adversarial test data.
  + Adversarial trained LR/vectorizer scores for the original test data and adversarial test data, when **k** = [1,2,3,4,5].
    - Note since there is randomness in the training process in how the adversarial examples are produced, run each model 5 times and take the average score (as you did in HW4). (You do not need to create a new adversarial test data every time, you can simply use the same one as long as it is consistent for all results).
  + Note, even though you will be running these functions multiple times, you do not need multiple instances of them.

**To be added to the report:**

* A table containing all of the produced results above in a good, readable format.
* Discussion on the results (You are encouraged to leverage graphs and other visuals to support your discussion):
  + Which model seems to be the best overall, and why.
  + Does adversarial training seem to be helping, why or why not.
  + If the training does not fully mitigate the attack, why might it be failing to do so?
  + How you might better improve the training to make the model more robust.

## To Turn In:

* Python file (**USERNAME\_EC1.py**.) which contains functions related to the above parts. You should include comments and useful function names to differentiate the testing portions.
* Report which contains:
  + The above requirements for each part. Make sure you make it clear (via titles or subheadings) which part you are describing.
  + Discussion of issues or difficulties encountered.
  + A note of any AI tools used in coding or writing.
* Your produced adversarial training file. (Even though you will be running the functions multiple times, you only need to hand in 1 (any k)).

**Additional Rules (MUST BE FOLLOWED):**

1. All Homeworks should follow the overall [homework guidelines](https://ind657-my.sharepoint.com/:p:/g/personal/jrusert_pfw_edu/EZxf1ZsRXjBEkQLNhEhdTgUBt6U64KiT1DJ1YHtkARgKwA?e=bwHrbf)
2. The code should be written in python 3.
3. If noted, the functions must follow the naming and number of arguments as demonstrated.
4. You should make your code modular to the different steps. (You may have more functions to help your main functions)
5. You should be adding comments to document your code. **If I can’t understand why you perform an action, then I can’t credit you for performing that action.**
6. The report should be readable and reference your code, **without explicitly including code.**
7. You should include your name and homework number in the comments at the beginning of the python file.

**Report**

The reports for the homeworks are necessary to communicate your learning and thinking through of the material. Examples of good reports can be found on brightspace under Additional Resources/Guides. Note that your report style may differ, but it is a good reference to start with.

**Grading**

Assignment will be graded as follows:

|  |  |
| --- | --- |
| **Description** | **Points** |
| **(.py file)** Adversarial training coding requirements | 10 |
| **(Report)** Adversarial training report requirements | 10 |
| **Total:** | **20** |

* **If the code does not run, it cannot be graded well.** (Many points can be lost if the code cannot be run, as I will not be able to fully test the implementation of the functions).
* **Breaking of the additional rules can result in applied penalties.** (Always make sure you are checking against the rules)

**Suggestions**

* **Documentation is key for showing your effort in this homework.** Make sure you are noting why you make certain decisions all throughout your code.
* The slides for previous classes are posted, so please refer to these and the book for ideas during implementation.
* Start simple, build up complexity. You should always make sure your new ideas being added do not cause your program to crash. So starting simple is the best way to a) maintain the ability to keep your code running, b) add in comments for documentation and thought process as you add more code.
* Work through the homework yourself, rather than sharing ideas (especially not code) with other students. **As a reminder, plagiarism (or sharing) of code is strictly prohibited.** This assignment is complex enough that significant overlap between students will be suspicious.
* If you have not worked with python before, w3schools can help you translate your previous coding experience to python (<https://www.w3schools.com/python/default.asp>)
* Stop by office hours to discuss ideas. I am always happy to help you think through your process!