COMP9414: Artificial Intelligence
Assignment 2: Rating Prediction

Due Date: Week 9, Wednesday, July 27, 11:59 p.m.

Value: 25%

This assignment is inspired by a typical real-life scenario. Imagine you have been hired as a Data Scientist by a major e-commerce retailer. Your job is to analyse customer reviews to determine whether you can predict the ratings of new products so they can be promoted on your website.

For this assignment, you will be given a collection of Amazon customer reviews. Each review consists of short text (a few sentences), and one of five *ratings*: a number from 1 to 5. You are required to evaluate various supervised machine learning methods using a variety of features and settings to determine what methods work best for rating prediction in this domain (these features could then be used to recommend items to users based on their interests).

The assignment has two components: *programming* to produce a collection of models for rating prediction, and a *report* to evaluate the effectiveness of the models. The programming part involves development of Python code for data preprocessing of reviews and experimentation of methods using NLP and machine learning toolkits. The report involves evaluating and comparing the models using various metrics.

You will use the NLTK toolkit for basic language preprocessing, and scikit-learn for feature construction and evaluating the machine learning models. You will be given an example of how to use NLTK and scikit-learn to define the machine learning methods (`example.py`), and an example of how to plot metrics in a graph (`plot.py`).

Data and Methods

A *training* dataset is a `.tsv` (tab separated values) file containing a number of reviews, with one review per line, and linebreaks within reviews removed. Each line of the `.tsv` file has three fields: instance_number, text and rating (a number from 1 to 5). A *test* dataset is a `.tsv` file in the same format as a training dataset *except that your code should ignore the rating field.* Training and test datasets can be drawn from supplied file `reviews.tsv` (see below). For evaluation of the models, we will use one 80–20 split of this file.

For all models, consider a review to be a collection of words, where a *word* is a string of at least two letters, numbers or the symbols / (slash), - (hyphen), $ or %, delimited by a space, *after* replacing two successive hyphens `--`, the tilde symbol ˜ and any ellipsis (three or more dots ...) by a space, then removing tags (minimal text spans between < and > inclusive) and all other characters. Two characters is the default minimum word length for CountVectorizer in scikit-learn. Note that deleting "junk" characters may create longer words that were previously separated by those characters, for example after removing tags, commas and full stops.

Use the supervised learning methods discussed in the lectures: Decision Trees (DT), Bernoulli Naive Bayes (BNB) and Multinomial Naive Bayes (MNB). Do not code these methods: instead use the implementations from scikit-learn. Read the scikit-learn documentation on Decision Trees[1] and Naive Bayes,[2] and the linked pages describing the parameters of the methods.

[1] https://scikit-learn.org/stable/modules/tree.html

[2] https://scikit-learn.org/stable/modules/naive_bayes.html

Look at `example.py` to see how to use CountVectorizer and train and test the machine learning algorithms, including how to generate metrics for the models developed, and `plot.py` to see how to plot these metrics on a graph for inclusion in your report.

The *programming* part of the assignment is to produce DT, BNB and MNB models and your own model for rating prediction in Python programs that can be called from the command line to train and classify reviews read from correctly formatted `.tsv` files. The *report* part of the assignment is to analyse these models using a variety of parameters, preprocessing tools and scenarios.

Programming

You will submit **four** Python programs: (i) `DT_classifier.py`, (ii) `BNB_classifier.py`, (iii) `MNB_classifier.py` and (iv) `my_classifier.py`. The first three of these are standard models as defined below. The last is a model that you develop following experimentation with the data. Use the given dataset `reviews.tsv` containing 2500 labelled reviews to develop and test the models, as described below.

These programs, when called from the command line, will read from standard input (**not** a hard-coded file `reviews.tsv`), and should print to standard output (**not** a hard-coded file `output.txt`), the instance_number and rating produced by the classifier of each review in the **test set** when trained on the training set (one per line with a space between the instance number and rating – a number from 1 to 5), where the training set is the first 80% of the file, and the test set is the remaining 20% (the file length will always be divisible by 5). For example:

```
python3 DT_classifier.py < reviews.tsv > output.txt
```

should write to the file `output.txt` the instance number and rating of each review in the test set (the last 20% of the file), as determined by the Decision Tree classifier trained on the training set (the first 80% of the file).

When reading in the dataset, make sure your code reads *all* the instances (some Python readers use “excel” format, which uses double quotes as separators).

Standard Models

You will develop three standard models. For all models, make sure that scikit-learn does **not** convert the text to lower case. For Decision Trees, use scikit-learn's Decision Tree method with criterion set to 'entropy' and with random_state=0. Scikit-learn's Decision Tree method does not implement pruning, rather you should ensure that Decision Tree construction stops when a node covers fewer than 1% of the training set. Decision Trees are prone to fragmentation, so to avoid overfitting and reduce computation time, for the Decision Tree models use as features only the 1000 most frequent words from the vocabulary, after preprocessing to remove “junk” characters as described above. Write code to train and test a Decision Tree model in `DT_classifier.py`.

For both BNB and MNB, use scikit-learn's implementations, but use *all* of the words in the vocabulary as features. Write two Pythons programs for training and testing Naive Bayes models, one a BNB model and one an MNB model, in `BNB_classifier.py` and `MNB_classifier.py`.

Your Model

Develop your best model for rating prediction by either varying the number and type of input features for the learners, the parameters of the learners, and the training/test set split, or by using another method from scikit-learn. Submit one program, `my_classifier.py`, that trains and tests a model in the same way as for the standard models. Conduct new experiments to analyse your model and present results that justify your choice of this model in the report.

Answer the questions properly

Report

In the report, you will first evaluate the standard models, then present your own model. For questions 1–4 below, consider two scenarios, where there are 5 classes in scenario 1 corresponding to the ratings, but 3 classes in scenario 2, where the ratings are combined into a "sentiment":

(1) the classes are the rating values from 1 to 5 (1, 2, 3, 4 and 5), and
(2) the classes are a "sentiment", where 1, 2 or 3 is *negative*, 4 is *neutral* and 5 is *positive*.

For evaluating all models, report the results of training on the first 2000 instances in the dataset (the "training set") and testing on the remaining 500 instances (the "test set").

Use the metrics (micro- and macro-accuracy, precision, recall and F1) and classification reports from scikit-learn. Show the results in Python plots (do **not** take screenshots of sklearn classification reports), and write a *short* response to each question below. The answer to each question should be self contained. **Your report should be at most 10 pages.** Do not include appendices.

1. (3 marks) Develop Decision Tree models for training and testing: (a) with the 1% stopping criterion (the standard model), and (b) without the 1% stopping criterion.

(i) Show all metrics on the test set for scenario 1 comparing the two models (a) and (b), and explain any similarities and differences.
(ii) Show all metrics on the test set for scenario 2 comparing the two models (a) and (b), and explain any similarities and differences.
(iii) Explain any differences in the results between scenarios 1 and 2.

2. (3 marks) Develop BNB and MNB models from the training set using: (a) the whole vocabulary (standard models), and (b) the most frequent 1000 words from the vocabulary, as defined using sklearn's CountVectorizer, after preprocessing by removing "junk" characters.

(i) Show all metrics on the test set for scenario 1 comparing the corresponding models (a) and (b), and explain any similarities and differences.
(ii) Show all metrics on the test set for scenario 2 comparing the corresponding models (a) and (b), and explain any similarities and differences.
(iii) Explain any differences in the results between scenarios 1 and 2.

3. (3 marks) Evaluate the effect of preprocessing for the three standard models by comparing models developed with: (a) only the preprocessing described above (standard models), and (b) applying, in addition, Porter stemming using NLTK then English stop word removal using sklearn's CountVectorizer.

(i) Show all metrics on the test set for scenario 1 comparing the corresponding models (a) and (b), and explain any similarities and differences.
(ii) Show all metrics on the test set for scenario 2 comparing the corresponding models (a) and (b), and explain any similarities and differences.
(iii) Explain any differences in the results between scenarios 1 and 2.

4. (3 marks) Evaluate the effect of converting all letters to lower case for the three standard models by comparing models with: (a) no conversion to lower case, and (b) all input text converted to lower case.

(i) Show all metrics on the test set for scenario 1 comparing the corresponding models (a) and (b), and explain any similarities and differences.
(ii) Show all metrics on the test set for scenario 2 comparing the corresponding models (a) and (b), and explain any similarities and differences.
(iii) Explain any differences in the results between scenarios 1 and 2.

5. (5 marks) Describe your chosen “best” method for rating prediction. Give **new** experimental results for your method trained on the training set of 2000 reviews and tested on the test set of 500 reviews. Explain how this experimental evaluation justifies your choice of model, including settings and parameters, against a range of alternatives. Provide **new** experiments and justifications: do not just refer to previous answers.

Submission

- **Make sure your name and zid appears on each page of the report**
- Submit all your files using a command such as (this includes Python code and report):

```
give cs9414 ass2 DT*.py BNB*.py MNB*.py my_classifier.py report.pdf
```

- Your submission should include:
  - Your `.py` files for the specified models and your model, plus any `.py` “helper” files
  - A `.pdf` file containing your report
- When your files are submitted, a test will be done to ensure that one of your Python files runs on the CSE machine **(take note of any error messages printed out)**
- When running your code on CSE machines:
  - Set `SKLEARN_SITE_JOBLIB=TRUE` to avoid warning messages
  - **Do not download NLTK in your code: CSE machines have NLTK installed**
- Check that your submission has been received using the command:

```
9414 classrun -check ass2
```

Assessment

Marks for this assignment are allocated as follows:

- Programming (auto-marked): 8 marks
- Report: 17 marks

Late penalty: Your mark is reduced by 1.25 marks per day or part-day late for up to 5 calendar days after the due date, after which a mark of 0 is given.

Assessment Criteria

- Correctness: Assessed on standard input tests, using calls such as:

```
python3 DT_classifier.py < reviews.tsv > output.txt
```

  Each such test will give one file which can contain **any** number of reviews (one on each line) in the correct format. The dataset can have any name, not just `reviews.tsv`, so read the file from standard input. The output should be a sequence of lines (one line for each review in the test set, i.e. the last 20% of the input file) giving the instance number and predicted rating, separated by a space and with no extra spaces on each line or extra newline characters following any newline character after the last prediction. There are 2 marks allocated for correctness of each of the three standard models.

  For your own method, 2 marks are allocated for correctness of your methods on datasets of reviews that include unseen examples.

- Report: Assessed on correctness and thoroughness of experimental analysis, clarity and succinctness of explanations, and presentation quality.

  There are 12 marks allocated to items 1–4 as above, and 5 marks for item 5. Of these 5 marks, 1 mark is for the description of your model, 2 marks are for **new** experimental analysis of your model, and 2 marks are for the justification of your model using **new** analysis. In general, if the presentation is of poor quality, at most 50% of the marks can be obtained.

Plagiarism

Remember that ALL work submitted for this assignment must be your own work and no sharing or copying of solutions (code or report) is allowed. You may use code from the Internet only with suitable attribution of the source in your program. **Do not use public code repositories on sites such as github – make sure your code repository, if you use one, is private.** All submitted assignments (both code and report) will be run through plagiarism detection software to detect similarities to other submissions, including from past years. Do not share your code or report with anyone both during **and after** the course has finished. You should **carefully** read the UNSW policy on academic integrity and plagiarism (linked from the course web page), noting, in particular, that *collusion* (working together on an assignment, or sharing parts of assignment solutions) is a form of plagiarism.

DO NOT USE ANY CODE FROM CONTRACT CHEATING "ACADEMIES" OR "TUTORING" SERVICES. THIS IS SERIOUS MISCONDUCT WITH A HEAVY PENALTY UP TO AUTOMATIC FAILURE OF THE COURSE WITH 0 MARKS.