Assignment 1
Sentiment Classification on Movie Reviews

STAT8021: Big Data Analytics (Spring 2025)
STAT8307: Natural Language Processing and Text Analysis (Spring 2025)

due: **March 2, 2025, Sunday, 11:59 PM**

Goals

The main goal of this assignment is to implement basic feature extraction and classification algorithms for text classification. After this, you will get a sense of how the machine learning pipeline works and how to design each component in this workflow. **Note that you should use Python 3.8+ to finish this assignment. Python machine learning packages (*e.g.*, Scikit-learn) are not allowed in this assignment.**

About Bonus: The total marks of this assignment is 100 (basic) + 20 (bonus) =120, *i.e.*, you can get at most $120 * 15\% = 18$ marks for this assignment in your final course marks, while the total marks of three assignments (A1-A3) will not exceed in 50 marks (total course marks are 100). If you have questions about this bonus part, please raise your questions in the Moodle forum.

Background

The Rotten Tomatoes movie review dataset is a corpus of movie reviews used for sentiment analysis, originally collected by Pang and Lee[1]. This dataset provides us a chance to benchmark our sentiment-analysis ideas. Its labels are "fine-grained" sentiment labels ranging from 0 to 4: negative, somewhat negative, neutral, somewhat positive, positive. In this assignment, we will tackle a simplified version of this problem: binary positive / negative sentiment classification. Positive labels are 1 and negative labels are 0. Obstacles like sentence negation, sarcasm, terseness, language ambiguity and many others make this task very challenging.

Data

You can find the required data by unzipping the `code.zip` file in Moodle and opening the `data` folder.

- `train.csv` contains the sentences and their associated sentiment labels for training.
- `valid.csv` contains the sentences and their associated sentiment labels for validation.
- `test.csv` contains just sentences. You must assign a sentiment label to each sentence.

Start Code

To help you solve this task, we provide you with a start code. **You only need to implement classes or functions with the hint "Write your code here". Please do not change other code**.

You can find the following five Python files after unzipping the `code.zip` file in Moodle:

- `main.py` is the main entry of this project.
- `metrics.py` contains classification metrics including accuracy, precision, recall and F-score.

[1] Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In ACL, pages 115–124.

- `feature_extractor.py` contains functions to extract features from raw text.
- `models.py` contains classification algorithms.
- `sentiment_data.py` contains functions to process raw data.

Submission

Please submit the following two files to Moodle for grading:

- A ZIP file of the following four Python files: `metrics.py`, `feature_extractor.py`, `models.py`, `sentiment_data.py`.
- A PDF report of your answers to the questions.
- **(Optional)** If you have finished the bonus question, please also submit the `test_predictions.csv` file in `data` folder.

The total score of this assignment is **100 (basic) + 20 (bonus) = 120 (total)**. Please write your procedures in the PDF report if you do not completely finish the code.

Part 1. Text preprocessing

Q1 This question requires you to implement basic text preprocessing methods. Specifically, please complete the `text_preprocessing` function in `sentiment_data.py`. You need to implement the following text preprocessing methods in order: NLTK tokenization, converting words into lower case, removing punctuations, removing stop words. Please run the following command and copy the output in your report. **[5 points]**

```
python sentiment_data.py
```

Part 2. Evaluation metrics

Q2 To evaluate the classification performance of your algorithm, you are required to implement the following metrics: accuracy, precision, recall and F-score. Here you should implement these metrics for multi-class classification problems. Thus, you also need to implement Macro-average and Micro-average methods to compute the overall scores. You can find the start code in `metrics.py`. Please complete this file. Also, you need to run the following command and copy the output into your report. **[20 points]**

```
python metrics.py
```

Q3 Now you are ready to run a trivial classifier to verify the correctness of your implemented metrics for binary classification. Please run the following command and copy the output into your report. Note that the correct output accuracy should be 0.5037, and the correct precision should be 0.5038. **[5 points]**

```
python main.py --model trivial
```

Part 3. Feature extraction

Q4 Bag-of-Words model (BoW) is a simplifying representation used in natural language and information retrieval. In this question, you are required to implement the Bag-of-Words model in Python. Specifically, you need to complete the `BoWFeatureExtractor` class in `feature_extractor.py`. After completing this file, please run the following command and copy the output into your report. **[20 points]**

```
python feature_extractor.py
```

Part 4. Text classification algorithms

Q5 Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on Bayes' theorem. In this question, you are required to implement the Naive Bayes algorithm. Please complete the `NaiveBayesClassifier` class in `models.py`. Then, run the following command and copy the output into your report. To get full marks, you have to get at least **85% accuracy** in the validation set. **[25 points]**

```
python main.py --model nb
```

Q6 Logistic regression is a classification algorithm to model the probability of a certain class or event. In this question, you are required to implement a logistic regression model using Stochastic Gradient Descent (SGD). Please complete the `LogisticRegressionClassifier` class in `models.py`. Then, run the following command and copy the output into your report. To get full marks, you should get at least **85% accuracy** in the validation set.
Note that we provide some backbone code in function `fit()` for your reference. You can follow this backbone or rewrite the whole function `fit()`.

[25 points]

```
python main.py --model lr
```

Part 5. Bonus

You may find the previous classification algorithms are suboptimal to this sentiment classification problem. In this part, you are encouraged to explore a better algorithm to solve this problem. Please write your plans about how to design a better algorithm in your report. Specifically, you should complete the `BetterFeatureExtractor` class in `feature_extractor.py` to design a better feature extractor and the `BetterClassifier` class in `models.py` to design a better classifier. Then, run the following command and copy the output into your report. In this question, you are also required to submit the `test_predictions.csv` file in `data` folder. If you need to change other parts of the given code to get improvement, please write your changes in your report and submit corresponding files.

[20 points]

```
python main.py --feats better --model better
```

Hint: Things you might try includes but are not limited to: n-grams feature (https://en.wikipedia.org/wiki/N-gram), tf-idf weighting, clipping your word frequencies, discarding rare words or stopwords, adding regularization terms in LR, using validation set to find better hyperparameters, or other variants of NB and LR classifier. For fair, please do not use deep neural networks:) **Your final code here should be whatever works best, and the model should train and evaluate in at most 20 mins on a SAAS/CS lab machine-equivalent computer.** The bonus score you will get depends on the relative performance on test data compared with other students' submission.