CSCI544: Homework Assignment №1

Due on Jan 30, 2026 (11:59 PM)

This assignment gives you hands-on experience with text representations and the use of text classification for sentiment analysis. Sentiment analysis is extensively used to study customer behaviors using reviews and survey responses, online and social media, and healthcare materials for marketing and customer service applications. The assignment includes a Jupyter Notebook to structure your code. Please submit:

1. A PDF report that contains answers to the questions in the assignment, along with brief explanations about your solution. Please also print the completed Jupyter Notebook in PDF format and merge it with your report. (Just submit one PDF file by merging your written answer and the completed Jupyter notebook.) In your completed Jupyter notebook, please also print the requested values.

2. You also need to submit an executable .py file, which, when run, generates the requested numerical outputs in the assignment as listed at the end of the assignment description. We need the .py file to check code overlaps and detect plagiarism. Please include the Python version you use.

The `HW1-CSCI544.ipynb` file includes the libraries you will need. You can use other libraries as long as they are decently similar to the ones included in the `HW1-CSCI544.ipynb` file. At the beginning of the .py file, add a read command to read the `data.tsv` file as the input to your .py file from the current directory.

For reproducibility, you must use `random_state = 42` whenever applicable (e.g., sampling data, train-test split, and model initialization).

1. Dataset Preparation (10 points)

We will use the Amazon reviews dataset, which contains real reviews for kitchen products sold on Amazon. The dataset is downloadable at:

`https://web.archive.org/web/20201127142707if_/https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Office_Products_v1_00.tsv.gz`

(a) Read the data as a Pandas frame using the Pandas package and only keep the Reviews (`review_body`) and Ratings (`star_rating`) fields in the input data frame to generate data. Our goal is to train sentiment analysis classifiers that can predict the sentiment (positive/negative) for a given review.

Include three sample reviews in your report along with corresponding ratings. Also, report the statistics of the ratings, i.e., how many reviews received 1 ratings, etc.

(b) We create binary labels using the ratings. We assume that ratings more than 3 demonstrate positive sentiment (mapped to 1) and rating less than or equal 2 demonstrate negative sentiment (mapped to 0). Discard reviews with a rating of 3 as neutral reviews. Include the number of reviews for each of these three classes in your report (to be printed by .py file in separate lines).

(c) The original dataset is large. To avoid computational burden, select 100,000 reviews with positive sentiment and 100,000 with negative sentiment to perform the required tasks on the downsized dataset. Split your dataset into 80% training dataset and 20% testing dataset. You can split your dataset after step 4 when the Bigram features are extracted.

Follow the given order of data processing, but you can change the order if it improves your final results.

2. Data Cleaning (20 points)

Implement the following steps to preprocess the dataset you created:
- convert all the reviews into lowercase.
- remove the HTML and URLs from the reviews
- remove non-alphabetical characters
- remove extra spaces

- perform contractions on the reviews, e.g., won't $\rightarrow$ will not. Include as many contractions in English as you can think of.

You can either use Pandas package functions or any other built-in functions. Do not try to implement the above processes manually. Most of the above processes can be performed with one line of code.

In your report, print the average length of the reviews in terms of character length in your dataset before and after cleaning (to be printed by .py file).

3. Preprocessing (20 points)

Use NLTK package to process your dataset:

- remove the stop words
- perform lemmatization

Print three sample reviews before and after data cleaning + preprocessing. In the .py file, print the average length of the reviews in terms of character length before and after preprocessing.

4. Feature Extraction (10 points)

Use NLTK package to extract Bigram features. At this point, you should have created a dataset that consists of features and binary labels for the reviews you selected.

5. Perceptron (10 points)

Train a Perceptron model on your training dataset using the sklearn built-in implementation. Report Accuracy, Precision, Recall, and f1-score on both the training and testing splits of your dataset. These 8 values should be printed in separate lines by the .py file.

6. SVM (10 points)

Train an SVM model on your training dataset using the sklearn built-in implementation. Report Accuracy, Precision, Recall, and f1-score on both the training and testing splits of your dataset. These 8 values should be printed in separate lines by the .py file.

7. Logistic Regression (10 points)

Train a Logistic Regression model on your training dataset using the sklearn built-in implementation. Report Accuracy, Precision, Recall, and f1-score on both the training and testing splits of your dataset. These 8 values should be printed in separate lines by the .py file.

8. Multinomial Naive Bayes (10 points)

Train a Multinomial Naive Bayes model on your training dataset using the sklearn built-in implementation. Report Accuracy, Precision, Recall, and f1-score on both the training and testing splits of your dataset. These 8 values should be printed in separate lines by the .py file.

To be consistent, when the .py file is run, the following should be printed:

- Statistics of three classes, formatted exactly as shown below (the values are examples):

  ```
  Positive reviews: 100000000
  Negative reviews: 100000000
  Neutral reviews: 100000000
  ```

- Average length of reviews before and after data cleaning, formatted to 4 decimal places (the values are examples):

  ```
  Average length before cleaning: 500.0000
  Average length after cleaning: 500.0000
  ```

- Average length of reviews before and after data preprocessing, formatted to 4 decimal places (the values are examples):

```
Average length before preprocessing: 500.0000
Average length after preprocessing: 500.0000
```

- Accuracy, Precision, Recall, and F1-score for training and testing split for Perceptron. Each of the 8 values must be on a new line and formatted to 4 decimal places. The format should look like this (the values are examples):

```
Perceptron Training Accuracy: 0.0000
Perceptron Training Precision: 0.0000
Perceptron Training Recall: 0.0000
Perceptron Training F1-score: 0.0000
Perceptron Testing Accuracy: 0.0000
Perceptron Testing Precision: 0.0000
Perceptron Testing Recall: 0.0000
Perceptron Testing F1-score: 0.0000
```

- Accuracy, Precision, Recall, and F1-score for training and testing split for SVM. Each of the 8 values must be on a new line and formatted to 4 decimal places. The format should look like this (the values are examples):

```
SVM Training Accuracy: 0.0000
SVM Training Precision: 0.0000
SVM Training Recall: 0.0000
SVM Training F1-score: 0.0000
SVM Testing Accuracy: 0.0000
SVM Testing Precision: 0.0000
SVM Testing Recall: 0.0000
SVM Testing F1-score: 0.0000
```

- Accuracy, Precision, Recall, and F1-score for training and testing split for Logistic Regression. Each of the 8 values must be on a new line and formatted to 4 decimal places. The format should look like this (the values are examples):

```
Logistic Regression Training Accuracy: 0.0000
Logistic Regression Training Precision: 0.0000
Logistic Regression Training Recall: 0.0000
Logistic Regression Training F1-score: 0.0000
Logistic Regression Testing Accuracy: 0.0000
Logistic Regression Testing Precision: 0.0000
Logistic Regression Testing Recall: 0.0000
Logistic Regression Testing F1-score: 0.0000
```

- Accuracy, Precision, Recall, and F1-score for training and testing split for Naive Bayes. Each of the 8 values must be on a new line and formatted to 4 decimal places. The format should look like this (the values are examples):

  ```
  Naive Bayes Training Accuracy: 0.0000
  Naive Bayes Training Precision: 0.0000
  Naive Bayes Training Recall: 0.0000
  Naive Bayes Training F1-score: 0.0000
  Naive Bayes Testing Accuracy: 0.0000
  Naive Bayes Testing Precision: 0.0000
  Naive Bayes Testing Recall: 0.0000
  Naive Bayes Testing F1-score: 0.0000
  ```

Note that in your Jupyter notebook, print the Accuracy, Precision, Recall, and f1-score for the above models in separate lines and in .py file in separate lines.