COL 774: Assignment 2 (Semester II, 2024-25)

Due Date: 11:59 pm, Wednesday March 17, 2025. Total Points: 48 + ___

Notes:

- This assignment has two main parts - text classification using Naïve Bayes (Part 1), and Classification using SVMs (Part II). Part I is being released now, and Part II will be released after the Minor exams. Both parts will be due at the same time.
- You should submit all your code (including any pre-processing scripts written by you) and any graphs that you might plot.
- Do not submit the datasets.
- Include a single write-up (pdf) file which includes a brief description for each question explaining what you did. Include any observations and/or plots required by the question in this single write-up file.
- You should use Python for all your programming solutions.
- Your code should have appropriate documentation for readability.
- You will be graded based on what you have submitted as well as your ability to explain your code.
- Refer to the course website for assignment submission instructions.
- This assignment is supposed to be done individually. You should carry out all the implementation by yourself.
- We plan to run Moss on the submissions. We will also include submissions from previous years since some of the questions may be repeated. Any cheating will result in a zero on the assignment, a penalty of -10 points and possibly much stricter penalties (including a fail grade and/or a DISCO).

1 (48 points) Text Classification using Naïve Bayes

In this problem, we will use the Naïve Bayes algorithm for text classification. The dataset for this problem is the AG News classification Dataset. Given a news description, the task is to predict the type of the news. There are 4 possible categories (labels) - World, Sports, Business and Science/Technology. You have been provided with the dataset, with the training and test splits containing 170,000 samples and 7,900 samples respectively. Data is available at this link.

In the AG News dataset, each article is associated with a title and a description. We will begin by exploring various standard techniques used in text classification to enhance model performance.

Since this dataset has two set of features (list of words in **title** and list of words in **description**), these methods will first be applied individually to a single set of feature (either the title or description). After identifying the best-performing models for each, we will combine their strengths to develop an optimal model that effectively utilizes both the title and description together.

1. (10 points) Implement the Naïve Bayes Multiclass classification algorithm to classify each sample into one of the given 4 categories. You should implement the model where for each word position in a document, we generate the word using a single (fixed across word positions) Multinoulli distribution.

   (a) Train the implemented Naïve Bayes Classifier using only the **description** text. Report the accuracy over the training as well as the test set.

   (b) Read about word cloud. Construct a word cloud representing the most frequent words for each class.

   Notes:

   - Make sure to use the Laplace smoothing for Naïve Bayes (as discussed in class) to avoid any zero probabilities.
   - You should implement your algorithm using logarithms to avoid underflow issues.
   - You should implement Naïve Bayes from the first principles and not use any existing Python modules.
   - We have provided a starter code class with a `fit()` method that takes in a pandas Dataframe containing a column with each entry containing a list of tokens. This class will be autograded.
   - It may be useful to write a "tokenizer" functions (e.g., a simple tokenizer is `.split()`) for various parts below, and simply call the above implemented class to fit models on the newly tokenized column.

2. (4 points) The dataset provided to you is in the raw format i.e., it has all the words appearing in the original set of articles. This includes words such as 'of', 'the', 'and' etc. (called stopwords). Presumably, these words may not be relevant for classification. In fact, their presence can sometimes hurt the performance of the classifier by introducing noise in the data. Similarly, the raw data treats different forms of the same word separately, e.g., 'eating' and 'eat' would be treated as separate words. Merging such variations into a single word is called stemming. Read about stopword removal and stemming (for text classification) online. **As earlier, you should perform this analysis on description text features.**

   (a) Perform stemming and remove the stop-words in the training as well as the validation data.

   (b) Construct word clouds for both classes on the transformed data.

   (c) Learn a new model on the transformed data. Report the validation set accuracy.

   (d) How does your accuracy change over the validation set? Comment on your observations.

3. (4 points) Feature engineering is an essential component of Machine Learning. It refers to the process of manipulating existing features/constructing new features in order to help improve the overall accuracy of the prediction task. In this part, we will use word based bi-grams as features.

Bigrams are word pairs created by combining two consecutive words in a sentence. For example, the phrase "Pizza is awfully good" would be tokenized into the following bigrams: ["Pizza is", "is awfully", "awfully good"]. Bigrams help capture contextual meaning, such as how the word "awfully" may have a negative connotation on its own but contributes to a positive sentiment in the phrase "awfully good."

Train a model that utilizes both unigrams (individual words) and bigrams as features, ensuring that preprocessing from part (2) is applied beforehand. After training, compare the model's performance in terms of training and test accuracy against the previous model to assess any improvements. **As earlier, you should perform this analysis on description text features.**

4. (2 points) Analyze the performance of different models to identify which one works best for classifying based on the description text (e.g., a unigram vs bigram model, model with/without stemming and/or stopword removal, etc.). Justify your selection using relevant performance metrics such as accuracy, precision, recall, F1-score, or any other evaluation criteria.

5. (10 points) Evaluating the Best Model for Title Features

   - Follow the same procedures outlined in parts 1, 2, and 3 & 4 above but this time only using the set of features (words) in **title**.
   - How does the accuracy obtained using best combination of title features compare with the accuracy obtained using (best combination of) description features? Comment on your observations.

6. (7 points) Now, we will explore how to develop models that incorporate both the title and description. Based on the best-performing model identified in the previous analysis, apply the same tokenization approach to these features. For example, if a bigram model without preprocessing from part (2) performed best for the description, tokenize the description using unigrams and bigrams from the raw text. Similarly, ensure that the title is tokenized according to the optimal approach determined earlier.

   (a) (3 points) To begin, we will ensure that our model learns the same set of parameters θ for both the title and description. This approach is equivalent to "concatenating" the two set of features into a single text representation and training our classifier on the merged text. After training, report the accuracies and compare with previous models using single set (title/description) of features.

   (b) (4 points) Now, we will allow the model to learn different parameters for title and description, $\theta^{(title)}$ and $\theta^{(desc)}$. Mathematically compute the best fit expression for these parameters using the maximum likelihood estimation (remember to include Laplace smoothing). Report the accuracies and compare with previous models using single set (title/description) of features. Also, how does the accuracy compare with a model using joint set of features, but using a simple concatenation (as in the part above)?

7. (3 points) Analyze the performance of your current best model compared to very simple baselines by performing the following steps:

   (a) What is the validation set accuracy that you would obtain by randomly guessing one of the categories as the target class for each of the articles (random prediction)?

   (b) What accuracy would you obtain if you simply predicted each sample as positive?

(c) How much improvement does your algorithm give over the random/positive baseline?

8. (3 points) Read about the confusion matrix. Explore the confusion matrix for the best model obtained so far:

   (a) Draw the confusion matrix for your best performing model (using both sets of features).

   (b) For each confusion matrix, which category has the highest value of the diagonal entry? What does that mean?

9. (5 points) As part of the feature engineering process, identify and create at least one additional set of features that could enhance your model's performance. Retrain the best-performing model obtained so far by including the newly engineered feature(s). Evaluate whether the inclusion of this feature leads to an improvement in accuracy. Compare the updated results with the previous model's performance and provide insights on the impact of the new feature.

2 Support Vector Machines

Coming soon.

Submission Instructions

You can find the starter code along with the data (downloaded from kaggle for the purpose of this assignment) at Assignment2_starter_code. The directory structure is given below. Your implementations will be autograded, and thus we request you to follow the following instructions while submitting:

- Note that some files will be added later (on the same link) for question 2.
- You need to implement the functions mentioned in the starter code files (e.g. in `naive_baiyes.py`). You may add any other functions or methods, but do not change the signature (name and arguments) of the given functions.
- You may add new files for the analysis (for plotting etc.) for each question in their respective folders. You may use a jupyter notebook or python file - whichever you prefer. Note that any and all code used by you (for plotting etc.) needs to submitted.
- While submitting, make sure you follow the same directory structure.

```
Assignment2/
├── data/
│   ├── Q1/
│   │   ├── train.csv
│   │   └── test.csv
│   └── Q2/
├── Q1/
│   └── naive_baiyes.py
└── Q2/
```