COL 774: Assignment 2 (Part A)

Due Date: 11:50 pm, Wednesday March 4, 2020. Total Points: 30 +

Notes:

- This assignment has two parts - Tweet Classification using Naïve Bayes (Part A) and Fashion MNIST Classification using SVM (Part B). 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. Do not submit any code that we have provided to you for processing.
- 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. **(33 points) Text Classification**
   In this problem, we will use the Naïve Bayes algorithm for classification of tweets by different twitter users. The dataset for this problem can be obtained from this website. Given a user's tweet, task is to predict the sentiment (Positive, Negative or Neutral) of the tweet. Read the website for more details about the dataset. The dataset contains separate training and test files containing 1.6 Million training samples and 498 test samples, respectively.

   (a) **(8 points)** Implement the Naïve Bayes algorithm to classify each of the tweets into one of the given categories. Report the accuracy over the training as well as the test set.
   Notes:
   - Make sure to use the Laplace smoothing for Naïve Bayes (as discussed in class) to avoid any zero probabilities. Use $c = 1$.
   - 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.

   In the remaining parts below, we will only worry about test accuracy.

   (b) **(2 points)** What is the test set accuracy that you would obtain by randomly guessing one of the categories as the target class for each of the review (random prediction). What accuracy would you obtain if you simply predicted the class which occurs most of the times in the training data (majority prediction)? How much improvement does your algorithm give over the random/majority baseline?

(c) **(3 points)** Read about the confusion matrix. Draw the confusion matrix for your results in the part (a) above (for the test data only). Which category has the highest value of the diagonal entry? What does that mean? What other observations can you draw from the confusion matrix? Include the confusion matrix in your submission and explain your observations.

(d) **(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 tweets. 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.
- Perform stemming and remove the stop-words in the training as well as the test data. You are free to use any libraries for this purpose.
- Remove Twitter username handles from the tweets. You can use nltk tokenizer package for this purpose. You are free to use any other libraries for this purpose also.
- Learn a new model on the transformed data. Again, report the accuracy.
- How does your accuracy change over test set? Comment on your observations.

(e) **(5 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 on the prediction task. For example, instead of using each word as a feature, you may treat bi-grams (two consecutive words) as a feature. Come up with at least two alternative features and learn a new model based on those features. Add them on top of your model obtained in part (d) above. Compare with the test set accuracy that you obtained in parts (a) and parts (d). Which features help you improve the overall accuracy? Comment on your observations.

(f) **(6 points)** TFIDF or tf–idf , short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. You can read more details about it from the link TF-IDF.

- Your task now is to use TF-IDF features with Gaussian Naive Bayes Model to predict the sentiments of the given tweets. You can use Scikit-Learn's TFIDFVectorizer and GaussianNB module for this purpose. How does your accuracy change over the test set? Comment on your observation.
- Selecting a smaller set of features is although not strictly necessary but it becomes handy when it is challenging to train a model with too many words or features. Use Scikit-learn's SelectPercentile to choose features with highest scores. Report your accuracy over the test set. Does selecting smaller set of features improves your time taken to train the model? Compare the time taken in both the cases. You can use python's time module for this purpose. Comment on your observation.

(g) **(2 points)** Receiver Operating Characteristic (ROC) metric is generally used to evaluate classifier output quality. ROC curves typically feature true positive rate on the Y axis, and false positive rate on the X axis. ROC curves are typically used in binary classification to study the output of a classifier. In order to extend ROC curve and ROC area to multi-label classification, it is necessary to binarize the output. One ROC curve can be drawn per label, but one can also draw a ROC curve by considering each element of the label indicator matrix as a binary prediction (micro-averaging). Another evaluation measure for multi-label classification is macro-averaging, which gives equal weight to the classification of each label. Read more about ROC curves from the link Link-1. Plot the suitable ROC curve as per the problem requirement and comment on your observation. You can use any available libraries of your choice for the same.

2. **Coming Soon..**