15. Develop a sentiment classifier for a kind of texts of your choice (e.g., tweets, product or movie reviews). Use an existing sentiment analysis dataset with at least two classes (e.g., positive/negative or positive/negative/neutral).[2] The classes should be mutually exclusive, i.e., this is a single-label multi-class classification problem. You may use Boolean, TF, or TF-IDF features corresponding to words or *n*-grams, to which you can also add other features (e.g., lexicon features). You may apply any feature selection (or dimensionality reduction) method you consider appropriate. You may also want to try using centroids of pre-trained word embeddings.[3] You can write your own code to produce feature vectors, perform feature selection (or dimensionality reduction) and train the classifier (e.g., using SGD, in the case of logistic regression), or you can use existing implementa[4]tions and software libraries.[5] You

[2] See, for example, the Large Movie Review Dataset (http://ai.stanford.edu/~amaas/data/sentiment/), the Cornell Movie Review Data (http://www.cs.cornell.edu/people/pabo/movie-review-data/, included in NLTK), or the Twitter Sentiment Analysis Dataset (https://www.kaggle.com/c/twitter-sentiment-analysis2/data, you need to create a Kaggle account).

[3] Pre-trained word embeddings are available, for example, from http://nlp.stanford.edu/projects/glove/, https://fasttext.cc/docs/en/crawl-vectors.html, https://code.google.com/archive/p/word2vec/.

should experiment with at least logistic regression (or multinomial logistic regression, if you have more than two classes) and optionally (if you are keen and have free time) additional learning algorithms (e.g., Naive Bayes, *k*-NN). Assume that each text is classified in the class the classifier considers most probable. Make sure that you use separate training, development, and test subsets. Tune the feature set and hyper-parameters (e.g., regularization weight λ in logistic regression) on the development subset. Include experimental results of a baseline majority classifier, i.e., a classifier that always assigns the most frequent class of the training data. Include in your report:

- Precision, recall, F1, precision-recall AUC scores, for each class and classifier, separately for the training, development, and test subsets. Use three separate tables for the training, development, and test results. In each table, use a separate row for each classifier (or baseline), and show the precision, recall, F1, PR-AUC scores of the classes in columns (four columns per class).
- Macro-averaged precision, recall, F1, precision-recall AUC scores (all computed by averaging the corresponding scores of the previous bullet over the classes), for each classifier, separately for the training, development, and test subsets.[6] Show these results by adding four more columns to the tables of the previous bullet.
- For each classifier, learning curves (slides 59, 62) showing macro-averaged F1 computed on (i) the training data the classifier has encountered, (ii) the entire development subset, (iii) the entire test subset. Show a separate diagram for each classifier, with three curves in each diagram.
- A short description of the methods and datasets you used, including statistics about the datasets (e.g., average document length, number of training/dev/test documents, vocabulary size) and a description of the preprocessing steps that you performed.

[5] Consider scikit-learn (http://scikit-learn.org/stable/), Weka (http://www.cs.waikato.ac.nz/ml/weka/), LIBSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm/).

[6] In *single-label* multi-class classification (often also called simply multi-class classification), micro-averaged precision, micro-averaged recall, and micro-averaged F1 are all equal to accuracy. Check https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1.