CSE 6240 Spr 2017 - Homework #1
Due: Feb 8th, 2017
Analyzing a Movie Review Dataset

Read through this tutorial on kaggle, https://www.kaggle.com/c/word2vec-nlp-tutorial/details/part-1-for-beginners-bag-of-words , to familiarize yourself with its python tools and workflow. Write your own annotated ipython notebook(s) to reproduce the steps in the blog and complete the exercises below. You can start with the sample code provided in the tutorial, but should clean it up, document and refactor as necessary.

1: Processing text to create design matrices
Prepare four design matrices, `X_counts, X_binary, X_tfidf, X_binary_imbalance` following the blog's procedure for removing markup, punctuation and stop words. `X_counts` entries contain the raw word counts (the blog does this). `X_binary` modifies `X_counts` so that all elements are either 0 or 1. `X_tfidf` modifies `X_counts` by applying the sklearn tfidf vectorizer (http://scikit-learn.org/stable/modules/feature_extraction.html) with `smooth_idf=false`. For the `X_binary_imbalance`, start with `X_counts`, set your rng seed to 0 and then delete 75% of the rows corresponding to `sentiment=1`. This creates an imbalanced data set.

2: Feature space similarity experiment
Write a function `dist(X, i, j, distance_function='Euclidean')` which returns the (Euclidean) distance between rows i and j of a design matrix. Also write a function `topk(X, k)` which returns `((i1,j1,d1),...(ik,jk,dk))` where `(ix,jx)` are the indices of the xth closest pair, and dx is the corresponding distance. You can break ties randomly. Then use `topk()` to find the closest review pairs for each design matrix and print the following: the indices of the reviews, the distance, the first 20 characters of each review, the labels for each review. Are the pairs always the same?

3. Classification Experiment
Now you're going to tune an SVM classifier using each design matrix, and measure the resultant performance. Read the sklearn docs (http://scikit-learn.org/stable/modules/cross_validation.html) on cross-validation to see the methods to use.

- Set your rng seed to 0 and create an initial learning_set / test_set split of 80-20.
- Now we want to use a linear SVM (`svm.SVC` with `kernel=linear`) and pick the best C value for our classifier
    - Repeat for each of the four design matrices:
        - Repeat 30 times:
            - Pick a random value of C uniformly in the interval (1e-4, 1e4)
            - Use 5-fold cross-validation to train the SVM

        - Estimate and record the ROC-AUC
    - Select the value of C which produced the best ROC-AUC measurement
    - Retrain the classifier using the entire learning set with this C value
    - Generate an ROC curve and annotate with the ROC-AUC
    - Submit test set predictions to Kaggle (see the section in the blog, and make sure you use their test data. You may need to retrain one more time using all "training data"). Add the Kaggle score to your ROC curve

Which design matrix performed best (e.g., which encoding method worked best)? What was the lift (improvement in AUC-ROC) between the worst and best cases for each experiment?

4. Learning Curve Experiment
Using a logistic regression classifier and the design matrix X_counts, generate a learning curve:

- Set your rng seed to 0 and create an initial learning_set / test_set split of 80-20
- Generate a learning curve (xval vs training error) for `n=(100, 500, 1000, 2000, 3000, 4000, 5000, 7500, 10000, 15000, 20000)` training instances
- Interpret the learning curve

Submit a single ipython notebook that contains clear, concise code and comments and all output requested.