**Shane Armstrong**

**CS 584 Team Name - H@ck3rm@n**

**Homework 3 – Text Clustering**

1. H@ck3rm@n
2. Rank – 7; Accuracy - 0.73

# Introduction:

Below are the following objectives of HW3:

<http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html#sphx-glr-auto-examples-cluster-plot-kmeans-silhouette-analysis-py>

1. Implement the K-Means Algorithm
2. Deal with Text Data (News Records) in Term-Document Sparse Matrix Format.
3. Design a Distance Function for Text Data
4. Think about Curse of Dimensionality
5. Think about Best Metrics for Evaluating Clustering Solutions.

General Approach and Data Mining Techniques

<http://scikit-learn.org/stable/auto_examples/bicluster/bicluster_newsgroups.html>

Application Architecture

Multiprocessing

Data processing

Data Validation

The V-measure is the harmonic mean between homogeneity and completeness.

Homogenous – the wrong clusters aren’t classified together

Completeness

Metrics

<http://scikit-learn.org/stable/modules/model_evaluation.html>

<http://scikit-learn.org/stable/modules/generated/sklearn.metrics.homogeneity_score.html>

<http://scikit-learn.org/stable/modules/generated/sklearn.metrics.completeness_score.html>

<http://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_measure_score.html>

<http://scikit-learn.org/stable/auto_examples/cluster/plot_adjusted_for_chance_measures.html>

Methodology

Next Steps

##### The Data

The data set is a sparse array of Boolean type parameters with the rows representing drugs and the columns representing parameters of those drugs. The data was given in a sparse format where the rows contained column numbers representing existence of that parameter for that drug.

##### Data Processing

I first tried to work with the data in a dense format when first using the data and it would take GBs of memory and caused most functions to crash in python. After a few experiments, my goal was to convert the data into a sparse array format to alleviate memory and computation constraints. I used the Python SciPy package to convert the text to a sparse arrays and Numpy’s module store those arrays to disk for later use. This convenient because of the built in functionality of SciPy’s package and because of the format of the data in the training and test files. I read in each row and created a list of jagged arrays (each drug had a different number of parameters and therefore their lengths varied). I converted those jagged arrays to a sparse matrix in COOrdinate (COO) format then to a Compressed Sparse Column (CSC) matrix. The COO matrix can be easily and efficiently converted to CSR or CSC matrices. Since I was dealing with column sparse data I decided to use CSC matrices to store the data. Numpy has a function to efficiently store sparse matrices which I used to store to disk.

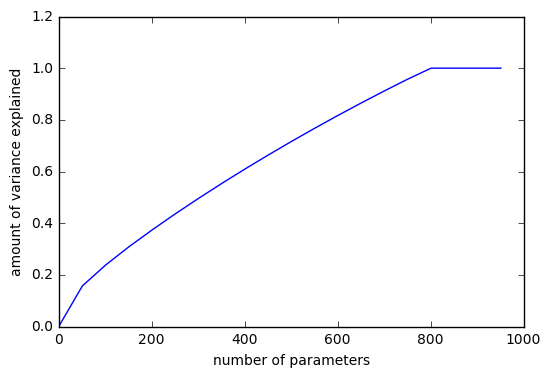
<http://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.coo_matrix.html>

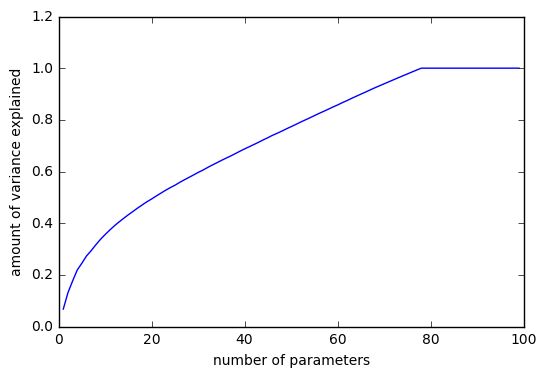
<http://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csc_matrix.html>

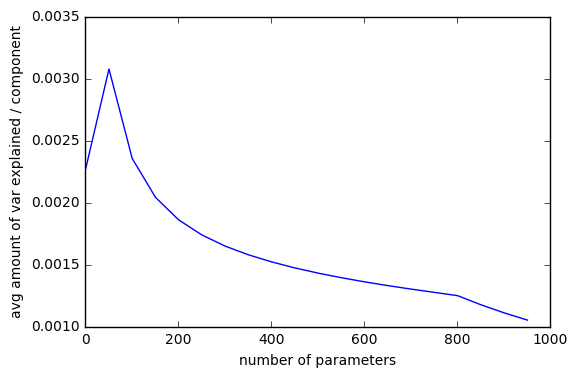
##### Data Validation

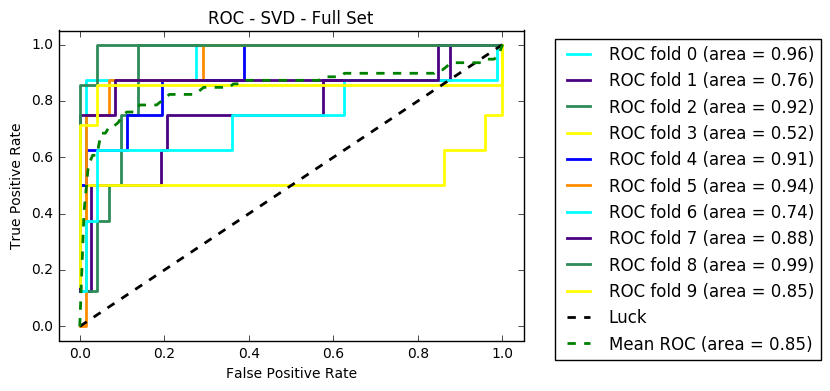
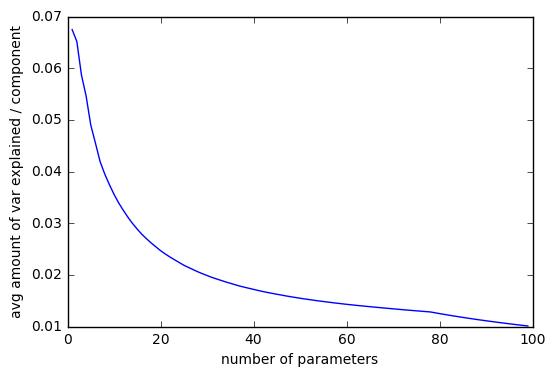
Data Validation was used to try different feature selection/dimension reduction and classification strategies. I relied heavily on Python’s scikit-learn module. My modeling decision were based on dealing with Boolean parameters, ability to be used with sparse matrices, and results from cross validation testing. For cross validation I sampled using Striated K-Folds methodology where k folds was varied and evaluated using a unweighted F1 score to emphasize precision and recall of positive classifications. My first set of runs was primarily for experimentation of how to use the scikit-learn module. I ran into several issues because of the use of sparse matrices. For instance scikit-learn’s implementation of principal component analysis (PCA) required dense array’s. Using PCA() with its automatic setting made the analysis of the sparse arrays impossible on my computer due to memory constraints. Other methodologies were necessary, such as using its randomized solver or using different implementations of PCA such as the IncrementalPCA. I performed dimension reduction with TruncatedSVD (Singular Value Decomposition) due to its ability to work with sparse arrays. For feature selection I used scikit-learn’s chi2 package. The statistic is more understandable and it makes sense to take out redundant parameters. I did not try to combine the two styles such as combining parameters with dimension reduction and then finding the most independent parameters with chi2. I am not sure that is sensible but something I’ll find out later. I created some visual aids for understanding the variance and cross validation error including a receiver operating characteristic (ROC) (code used from scikit-learn example)

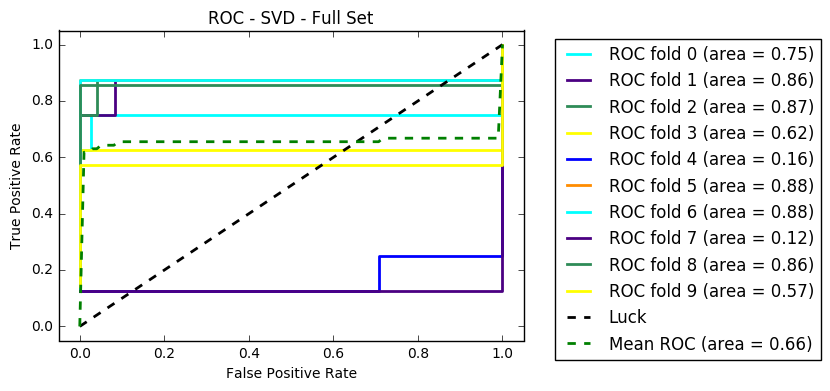
CV with SVD on positive data set CV on SVD with full data set



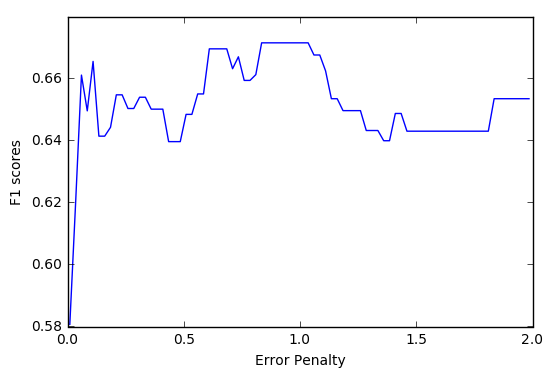








F1 score with different SVM error penalties



Cross Validation:

<http://scikit-learn.org/stable/modules/cross_validation.html>

<http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html>

<http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html>

Dimension Reduction/Feature Selection:

<http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>

<http://scikit-learn.org/stable/modules/classes.html>

<http://scikit-learn.org/stable/modules/feature_selection.html>

<http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html>

For ROC:

<http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc_crossval.html#sphx-glr-auto-examples-model-selection-plot-roc-crossval-py>

<http://scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.html>

# Methodology

To classify the data I tried using three different classification methodologies, linear support vector machines (SVM), Bernoulli naïve bayes, and k nearest neighbor (kNN). The SVM classification performed the best without performing any data processing with an ~.50 score. Naïve bayes scored .11 on the test data with no data processing. Naïve bayes performed substantially better with feature selection using chi2. SVM performed best using SVD. The best mean f1 score with cross validation for naïve bayes with chi2 feature selection was .7562 with 350 features from the full data set, and the best mean f1 score for SVM with SVD dimension reduction was 0.878 with 21 features and an error of 1 training on the positive data set. I added kNN last. algorithm did not perform well under the circumstances I tested including parameter reduction/feature selection, k changes, test sets used to train, the best I was getting was ~.60 with cross validation. I tried to use the prediction values from naïve bayes and SVM for calculating neighbors for the kNN classifier but it severly over fitted, showing an f1 score of ~.98 but test at ~.20. I could see a few reasons for this including the sensitivity of the predictions and I did not do much research into correcting for the overfitting. I predicted the test set by combining my naïve bayes and SVM predictions. If either predicted it was a positive I assumed it was a positive. I planned on using the kNN predictor as a tie breaker but it is unreliable averages approximately

Classifiers:

<http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html>

<http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

<http://scikit-learn.org/stable/modules/naive_bayes.html>

<http://scikit-learn.org/stable/modules/svm.html>

<http://scikit-learn.org/stable/modules/neighbors.html>