Machine Learning

CS-583 DMTM Project 2: Aspect-Sentiment Analysis (Reviews dataset. 2 nos.)

# Phase 1

## Stage I

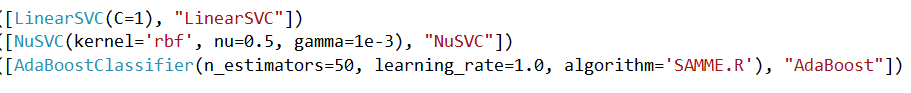
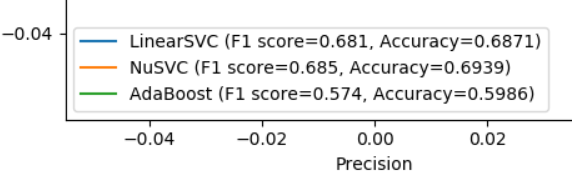
1. Loaded data using Pandas.read\_data()
2. In Preprocessing
   1. Throw away punctuations.
   2. Use SnowBall stemmer with stopwords: English.
   3. TODO: consider pickle dumping (for offline loading of pre-processed data).
   4. TF-IDF Vectorization
      1. TODO: Use 2-grams. (DONE. Visualizations added)
3. Resolved issue with multiclass F1. Used f1-score(average=’weighted’)
4. Issue with precision\_recall\_curve() not supporting multiclass.
5. Used only ‘document’ and ‘label’ columns for now.
6. Used pyplot to write plotting library.
7. Initial run o/p

Processing 2203 samples with 5 attributes  
Evaluating classifiers

Plotting the results

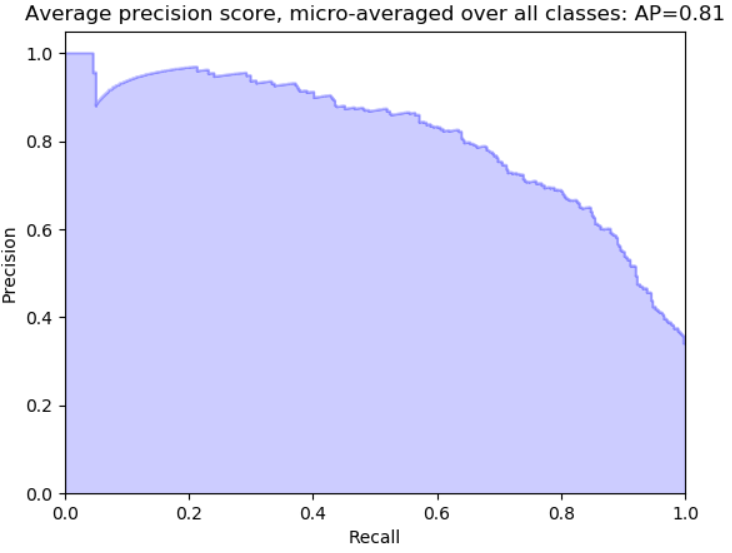
Accuracy of clf: Linear SVC (F1 score=0.721) = 0.7188208616780045

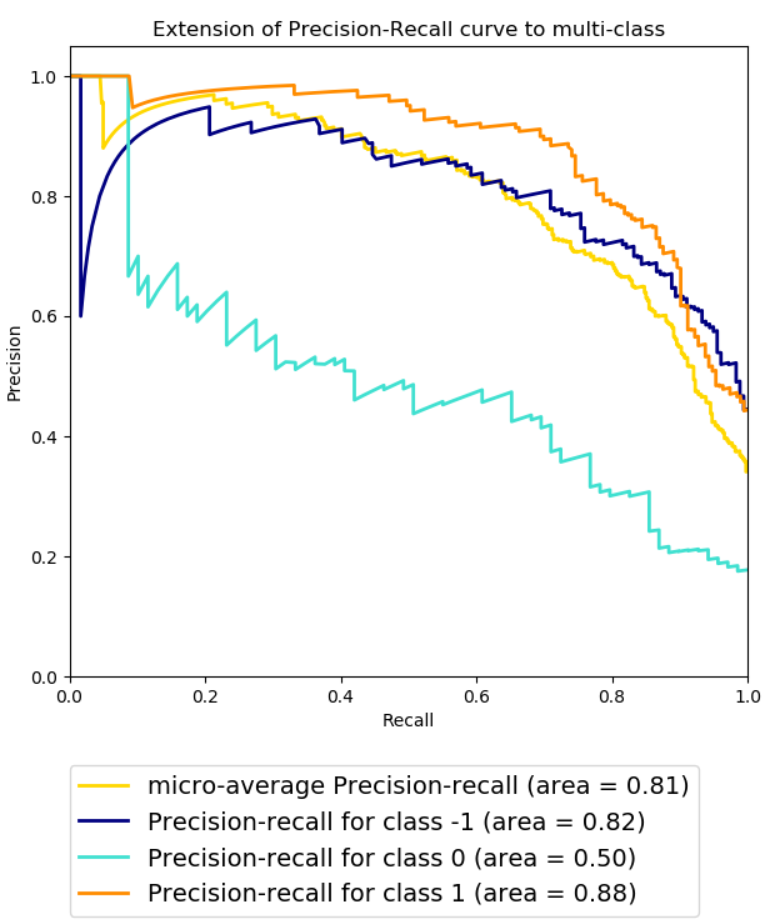
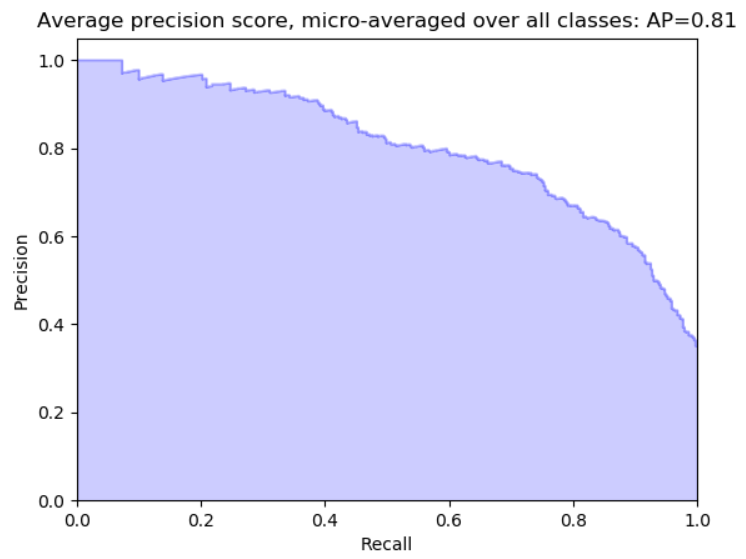
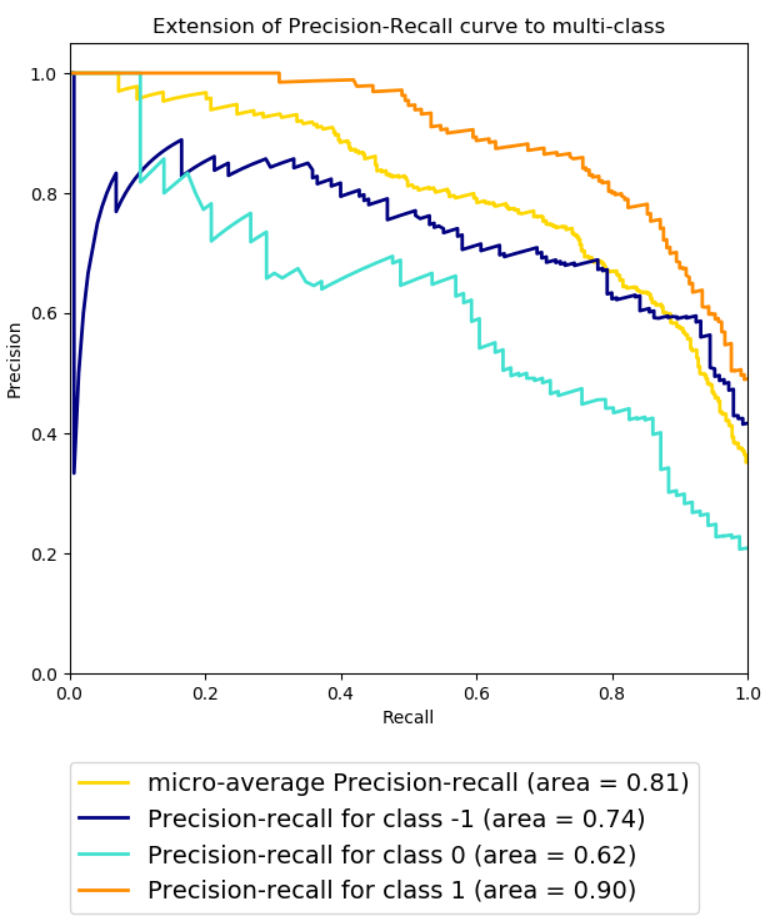
Accuracy of clf: NuSVC (F1 score=0.729) = 0.7301587301587301

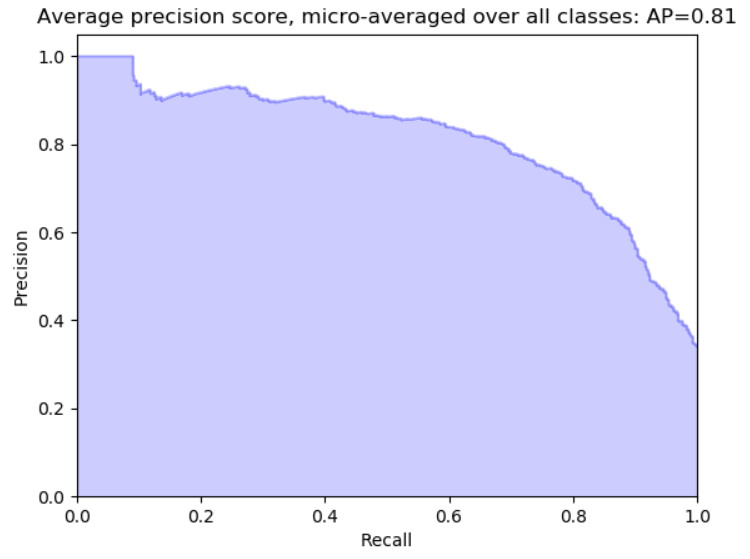
Accuracy of clf: Ada Boost (F1 score=0.589) = 0.6009070294784581  
  
  
  


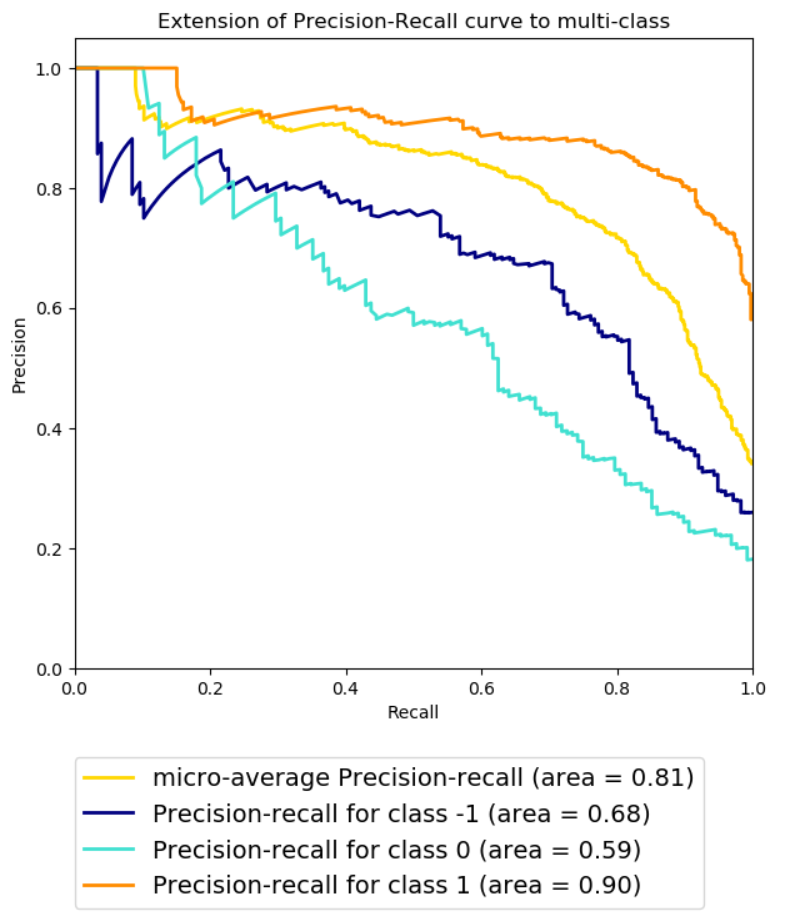
## Stage II

* 1. Implemented multiclass Pre/Re
     1. Used label binarizer
     2. Used OVR with LinearSVC (default params)
     3. Used Micro-Averaged PR
        1. Micro- and macro-averages (for whatever metric) will compute slightly different things, and thus their interpretation differs. A macro-average will compute the metric independently for each class and then take the average (hence treating all classes equally), whereas a micro-average will aggregate the contributions of all classes to compute the average metric. In a multi-class classification setup, micro-average is preferable if you suspect there might be class imbalance (i.e you may have many more examples of one class than of other classes).
        2. Ref: <https://datascience.stackexchange.com/questions/15989/micro-average-vs-macro-average-performance-in-a-multiclass-classification-settin/16001>
  2. CV
     1. Single (80-20) fold for now.
     2. TODO: 10-fold CV
  3. Uses average weighted F1, and overall accuracy.
  4. o/p  
     **OVR LinearSVC (F1 score=0.725, Accuracy=0.6644)**

**Average precision score, micro-averaged over all classes: 0.81**  
  


* 1. 
  2. A comparison w/o Stemmer
     1. **OVR LinearSVC (F1 score=0.719, Accuracy=0.6667)**
     2. **Average precision score, micro-averaged over all classes: 0.81**
  3.   
       
     
  4. o/p from data 2 (with stemming)  
     **OVR LinearSVC (F1 score=0.725, Accuracy=0.6907)**

**Average precision score, micro-averaged over all classes: 0.81**  
  


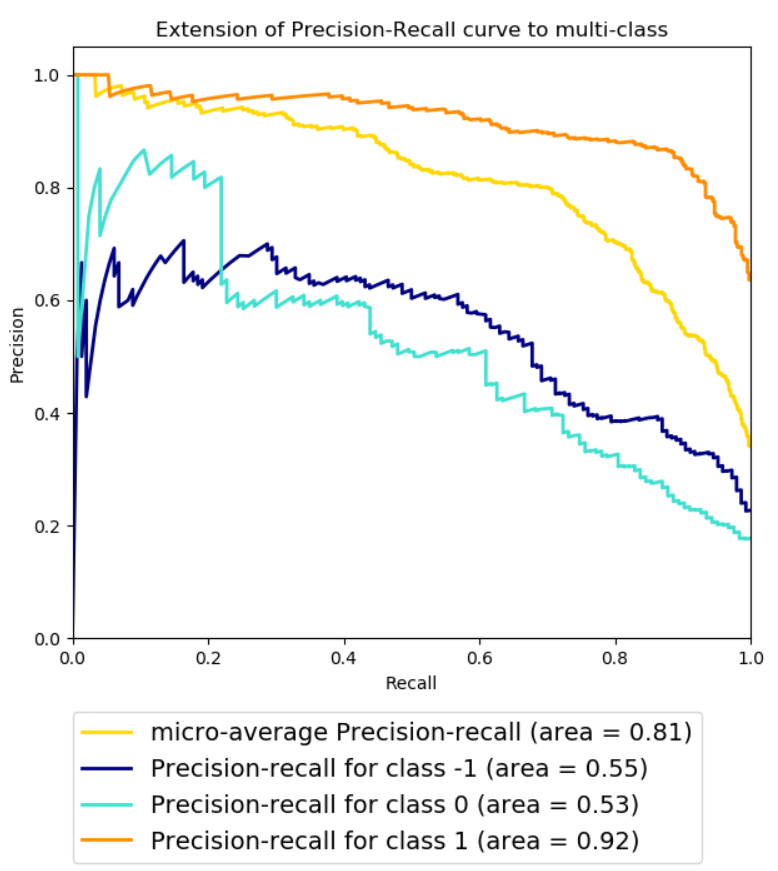


* 1. Evaluations using uni+bi-gram vectorizer (data 2)

**Evaluating classifiers**

**OVR LinearSVC (F1 score=0.731, Accuracy=0.7060)**

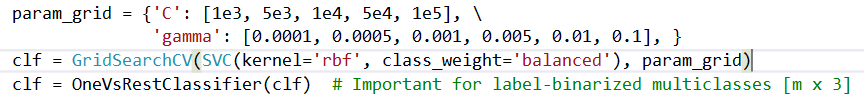
**Average precision score, micro-averaged over all classes: 0.81**



* 1. TODO:
     1. Sample data visualization
     2. Feature engineering
        1. Subject-predicate identification (Target-aspect).
        2. Context-sensitive weighting vectorizer.
     3. Outlier removal
     4. Scaling (? Is this really relevant in text classification?)
        1. Centering
        2. Min-Max
        3. PCA (reduction)
     5. More classifiers (KNN, NB, SVC (poly/rbf), RandomForest)
     6. Hyper-parameter tuning
        1. GridSearch
        2. Manual
     7. NN / Deep Learning

1. Ref: <http://scikit-learn.org/stable/auto_examples/text/document_classification_20newsgroups.html#sphx-glr-auto-examples-text-document-classification-20newsgroups-py>

# Phase 2

1. Evaluation results using RBF SVM with GridSearch (Data 2):  
     
   

**Processing 3602 samples with 5 attributes**

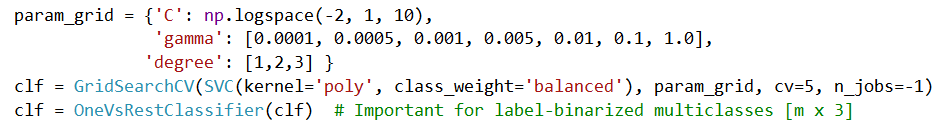
**Evaluating classifiers**

**SVC with RBF and GridSearchCV (F1 score=0.725, Accuracy=0.6782)**

**Average precision score, micro-averaged over all classes: 0.72**

No visible improvement!

1. Using 5-Fold CV (GridSearchCV) with SVM poly kernel and OVR multiclass discrimination.

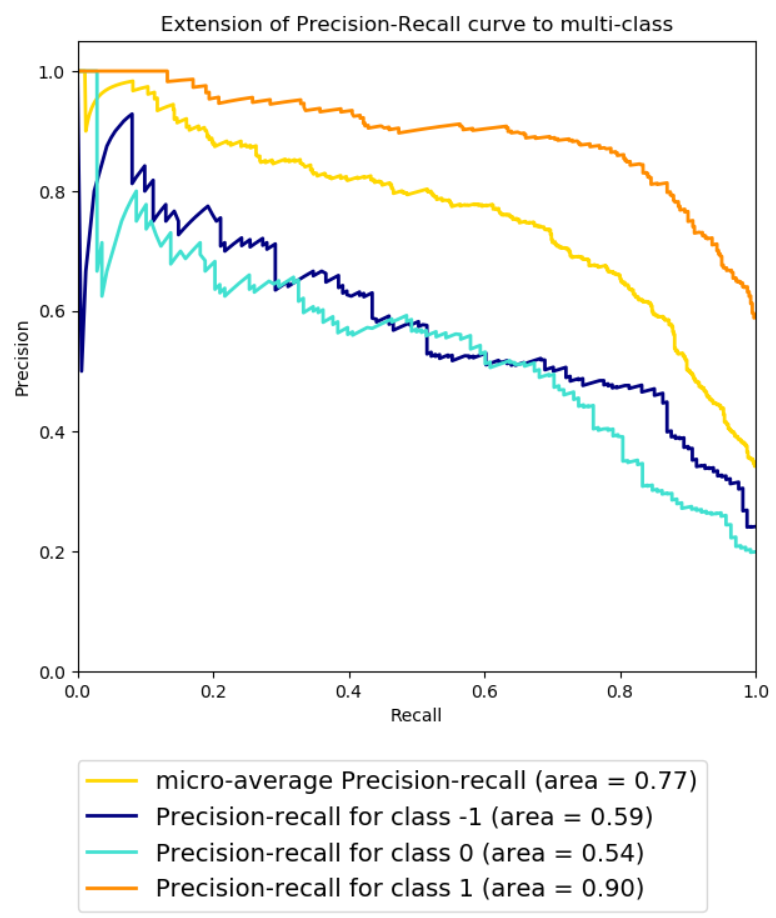
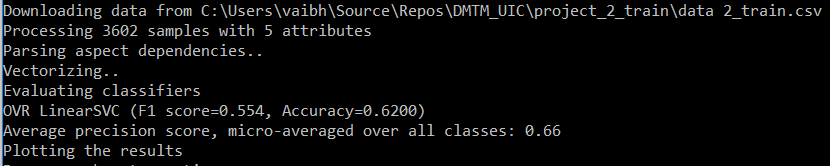
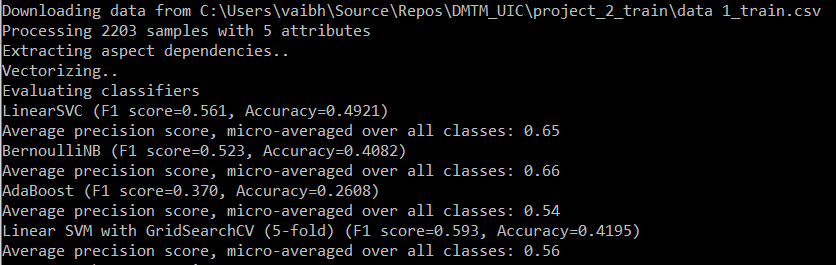
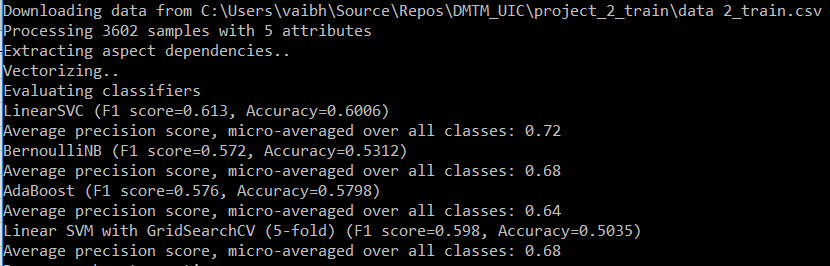
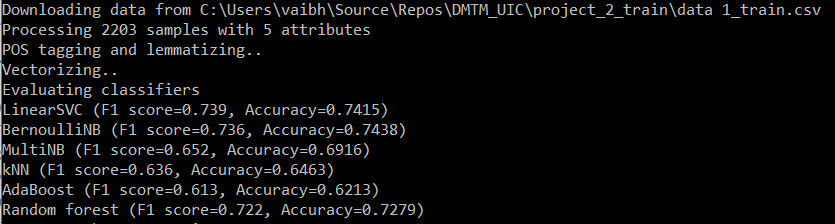


**Processing 3602 samples with 5 attributes**

**Evaluating classifiers**

**poly SVM with OVR and GridSearchCV (F1 score=0.675, Accuracy=0.6366)**

**Average precision score, micro-averaged over all classes: 0.72**

1. Using 5-Fold CV (GridSearchCV) with Linear SVM and OVR multiclass discrimination.  
   **Linear SVM with OVR and GridSearchCV=5 (F1 score=0.717, Accuracy=0.6436)**  
     
   
2. Researching NLTK (includes Stanford NLP parser)
   1. Parsed recursive dependencies using scoped dependency-type list.
   2. First o/p  
        
      
   3. Further scoped to extract pos - adjectives and nouns.
   4. Excluded stop-words.  
      o/p  
        
        
      
3. TODO:
   1. Check why data1 accuracy is lower than data2.
   2. Try WordToVec, research best methods, Deep Learning
4. Using another text processing strategy: Document->Sentences->Tokens->POS->Lemmas
   1. Used NLTK pos tagger and lemmatizer
5. Results:
   1. Data1: **Accuracy 74%, using LinearSVC and BernoulliNB.**
   2. Data2: **Accuracy 71%, using LinearSVC (C=5)**  
      