Apply SVM to Amazon reviews data set [M]

June 1, 2018

```
In [2]: !pip install --user imblearn

Collecting imblearn
  Using cached https://files.pythonhosted.org/packages/81/a7/4179e6ebfd654bd0eac0b9c06125b8b4c96
Collecting imbalanced-learn (from imblearn)
  Using cached https://files.pythonhosted.org/packages/80/a4/900463a3c0af082aed9c5a43f4ec317a946
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.6/dist-packages (from imba
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from imbalanced-
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from imbalanced-
Installing collected packages: imbalanced-learn, imblearn
Successfully installed imbalanced-learn-0.3.3 imblearn-0.0
You are using pip version 9.0.1, however version 10.0.1 is available.You should consider upgradi
```

```
In [3]: from sklearn.model_selection import train_test_split

        from sklearn.grid_search import GridSearchCV
        from sklearn.grid_search import RandomizedSearchCV
        from scipy.stats import uniform
        from scipy.stats import norm

        from imblearn.over_sampling import SMOTE

        import sqlite3

        import pandas as pd
        import numpy as np

        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.feature_extraction.text import CountVectorizer
        import gensim

        from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

        from sklearn.preprocessing import StandardScaler

/home/vishal/anaconda3/lib/python3.6/site-packages/sklearn/cross_validation.py:41: DeprecationWa
  "This module will be removed in 0.20.", DeprecationWarning)
```

```
/home/vishal/anaconda3/lib/python3.6/site-packages/sklearn/grid_search.py:42: DeprecationWarning
  DeprecationWarning)
```

```
In [17]: con = sqlite3.connect('final.sqlite') # this is cleaned dataset
         final = pd.read_sql_query("""
         SELECT Score, Text_not_included
         FROM reviews
         """, con)[:2000]
```

```
In [18]: for i, seq in enumerate(final['Text_not_included']):
           final['Text_not_included'][i]=final['Text_not_included'][i].decode('UTF-8')
         X_train, X_test, y_train , y_test = train_test_split(final['Text_not_included'], final[
```

0.1 Generate TF IDF weighted word2Vec

```
In [19]: tf_idf_vect=TfidfVectorizer(ngram_range=(1,2), min_df=10, dtype=float)
         tf_idf_vect.fit(X_train)
         tf_idf_train=tf_idf_vect.transform(X_train)
         tf_idf_test=tf_idf_vect.transform(X_test)
```

```
In [20]: w2vec_model=gensim.models.word2vec.Word2Vec(sentences, min_count=10)

         avg_w2vec_train=np.zeros(shape=(len(X_train), 100), dtype=float)
```

```
In [21]: sentences=[]
         for review in X_train:
           sentence=[]
           for word in review.split():
             sentence.append(word)
           sentences.append(sentence)

         tf_idf_w2vec_train=np.zeros((len(X_train), 100), dtype=float)
         feat=tf_idf_vect.get_feature_names()
         for i, sentence in enumerate(sentences):
           tf_idf_sum=0
           for word in sentence:
             try:
               tf_idf_w2vec_train[i]+=w2vec_model.wv[word]*tf_idf_train[i, feat.index(word)]
               tf_idf_sum+=tf_idf_train[i, feat.index(word)]
             except KeyError:
               pass
             except ValueError:
               pass
           tf_idf_w2vec_train[i]/=tf_idf_sum

         sentences=[]
         for review in X_test:
           sentence=[]
```

```
  for word in review.split():
    sentence.append(word)
  sentences.append(sentence)

tf_idf_w2vec_test=np.zeros((len(X_test), 100), dtype=float)

for i, sentence in enumerate(sentences):
  tf_idf_sum=0
  for word in sentence:
    try:
      tf_idf_w2vec_test[i]+=w2vec_model.wv[word]*tf_idf_test[i, feat.index(word)]
      tf_idf_sum+=tf_idf_test[i, feat.index(word)]
    except KeyError:
      pass
    except ValueError:
      pass
  tf_idf_w2vec_test[i]/=tf_idf_sum
```

```
In [22]: # Upsampling minority class
         over_sampler = SMOTE(ratio='minority')

         tf_idf_train_resampled, y_train_resampled = over_sampler.fit_sample(tf_idf_train, y_tra

         tf_idf_w2vec_train_resampled, y_train_resampled = over_sampler.fit_sample(tf_idf_w2vec_

         scaler_tf_idf=StandardScaler(with_mean=False)

         scaler_tf_w2vec=StandardScaler()


         scaler_tf_idf.fit(tf_idf_train_resampled)

         scaler_tf_w2vec.fit(tf_idf_w2vec_train_resampled)


         tf_idf_train_scaled=scaler_tf_idf.transform(tf_idf_train_resampled)

         tf_idf_w2vec_train_scaled=scaler_tf_w2vec.transform(tf_idf_w2vec_train_resampled)


         tf_idf_test_scaled=scaler_tf_idf.transform(tf_idf_test)

         tf_idf_w2vec_test_scaled=scaler_tf_w2vec.transform(tf_idf_w2vec_test)
```

```
In [23]: from sklearn.svm import SVC
```

```
In [32]: tuned_parameters = {'C': np.linspace(10, 20, 20, dtype=float),'gamma' : np.linspace(0.0
#Using GridSearchCV
gscv = GridSearchCV(SVC(), tuned_parameters, scoring = 'accuracy', cv=5)

print(gscv.fit(tf_idf_w2vec_train_scaled, y_train_resampled))

tuned_parameters = {'C' : uniform(10,20), 'gamma' : uniform(0, 1)}

#Using RandomizedSearchCV
rscv = RandomizedSearchCV(SVC(), tuned_parameters, scoring = 'accuracy', cv=5, n_iter=1

print(rscv.fit(tf_idf_w2vec_train_scaled, y_train_resampled))
```

```
GridSearchCV(cv=5, error_score='raise',
       estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False),
       fit_params={}, iid=True, n_jobs=1,
       param_grid={'C': array([ 10.      ,  10.52632,  11.05263,  11.57895,  12.10526,  12.63158,
        13.15789,  13.68421,  14.21053,  14.73684,  15.26316,  15.78947,
        16.31579,  16.84211,  17.36842,  17.89474,  18.42105,  18.94737,
        19.47368,  20.     ]), 'gamma': array([ 0.001   ,  0.0535...937,
        0.63195,  0.68453,  0.73711,  0.78968,  0.84226,  0.89484,
        0.94742,  1.     ])},
       pre_dispatch='2*n_jobs', refit=True, scoring='accuracy', verbose=0)
RandomizedSearchCV(cv=5, error_score='raise',
          estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False),
          fit_params={}, iid=True, n_iter=10, n_jobs=1,
          param_distributions={'C': <scipy.stats._distn_infrastructure.rv_frozen object at 0x7fe
          pre_dispatch='2*n_jobs', random_state=None, refit=True,
          scoring='accuracy', verbose=0)
```

In [33]:

```
predictions = gscv.best_estimator_.predict(tf_idf_w2vec_test_scaled)
print(classification_report(y_test, predictions))
print(confusion_matrix(y_test, predictions).T)
tn, fp, fn, tp = confusion_matrix(y_test, predictions).ravel()

print("TPR = {}\n TNR = {}\n FPR = {}\n FNR = {}".format(tp/(fn+tp), tn/(tn+fp), fp/(tn

predictions = rscv.best_estimator_.predict(tf_idf_w2vec_test_scaled)
print(classification_report(y_test, predictions))
print(confusion_matrix(y_test, predictions).T)
tn, fp, fn, tp = confusion_matrix(y_test, predictions).ravel()

print("TPR = {}\n TNR = {}\n FPR = {}\n FNR = {}".format(tp/(fn+tp), tn/(tn+fp), fp/(tn
             precision    recall  f1-score   support

   negative       0.57      0.05      0.09        87
   positive       0.79      0.99      0.88       313

avg / total       0.74      0.79      0.71       400

[[  4   3]
 [ 83 310]]
TPR = 0.9904153354632588
 TNR = 0.04597701149425287
 FPR = 0.9540229885057471
 FNR = 0.009584664536741214
             precision    recall  f1-score   support

   negative       0.21      0.03      0.06        87
   positive       0.78      0.96      0.86       313

avg / total       0.66      0.76      0.69       400

[[  3  11]
 [ 84 302]]
TPR = 0.9648562300319489
 TNR = 0.034482758620689655
 FPR = 0.9655172413793104
 FNR = 0.03514376996805112
```

```
In [34]: gscv.best_estimator_
```

```
Out[34]: SVC(C=10.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma=0.631947368421052559,
           kernel='rbf', max_iter=-1, probability=False, random_state=None,
           shrinking=True, tol=0.001, verbose=False)
```

```
In [35]: rscv.best_estimator_
```

```
Out[35]: SVC(C=20.582199592274449, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma=0.49642755839081232,
           kernel='rbf', max_iter=-1, probability=False, random_state=None,
           shrinking=True, tol=0.001, verbose=False)
```

0.2 Conclusion

SVMs take a lot of time to train even on small datasets with less dimensions. Best gamma is about 0.5 TNR as found by GSCV as well as RSCV is very poor C can be taken to be between 10 and 20