svm

November 9, 2019

0.1 Support Vector Machine model

```
In [ ]: #### Import necessary libraries
        import pandas as pd
        from sklearn import metrics, svm
        from sklearn.model_selection import train_test_split, GridSearchCV
        from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
        from sklearn.calibration import CalibratedClassifierCV
        import pickle
```

```
In [2]: # read the first 500,000 yelp reviews
        # df = pd.read_json('yelp_dataset/review.json', lines = True)
        # df = df[0:500000]
        df = pd.read_csv("yelp_dataset/yelp_reviews.csv", encoding='utf-8')
```

```
In [3]: df.head(5)
```

```
Out[3]:                business_id  cool                 date  funny  \
        0  ujmEBvifdJM6h6RLv4wQIg     0  2013-05-07 04:34:36      1
        1  NZnhc2sEQy3RmzKTZnqtwQ     0  2017-01-14 21:30:33      0
        2  WTqjgwHlXbSFevF32_DJVw     0  2016-11-09 20:09:03      0
        3  ikCg8xy5JIg_NGPx-MSIDA     0  2018-01-09 20:56:38      0
        4  b1b1eb3uo-w561D0ZfCEiQ     0  2018-01-30 23:07:38      0

                        review_id  stars  \
        0  Q1sbwvVQXV2734tPgoKj4Q      1
        1  GJXCdrto3ASJ0qKeVWPi6Q      5
        2  2TzJjDVDEuAW6MR5Vuc1ug      5
        3  yi0R0Ugj_xUx_Nek0-_Qig      5
        4  11a8sVPMUFtaC7_ABRkmtw      1

                                                        text  useful  \
        0  Total bill for this horrible service? Over $8G...       6
        1  I *adore* Travis at the Hard Rock's new Kelly ...       0
        2  I have to say that this office really has it t...       3
        3  Went in for a lunch. Steak sandwich was delici...       0
        4  Today was my second out of three sessions I ha...       7
```

```
                 user_id
0  hG7b0MtEbXx5QzbzE6C_VA
1  yXQM5uF2jS6es16SJzNHfg
2  n6-Gk65cPZL6Uz8qRm3NYw
3  dacAIZ6fTM6mqwW5uxkskg
4  ssoyf2_x0EQMed6fgHeMyQ
```

```
In [4]: df.describe()
```

```
Out[4]:                  cool          funny          stars         useful
        count  500000.000000  500000.000000  500000.000000  500000.000000
        mean        0.551726       0.453300       3.729382       1.307716
        std         2.035998       1.679424       1.455030       2.979647
        min         0.000000       0.000000       1.000000       0.000000
        25%         0.000000       0.000000       3.000000       0.000000
        50%         0.000000       0.000000       4.000000       0.000000
        75%         0.000000       0.000000       5.000000       1.000000
        max       203.000000     146.000000       5.000000     201.000000
```

```
In [5]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500000 entries, 0 to 499999
Data columns (total 9 columns):
business_id    500000 non-null object
cool           500000 non-null int64
date           500000 non-null object
funny          500000 non-null int64
review_id      500000 non-null object
stars          500000 non-null int64
text           500000 non-null object
useful         500000 non-null int64
user_id        500000 non-null object
dtypes: int64(4), object(5)
memory usage: 34.3+ MB
```

```
In [3]: # split the dataset into training and validation datasets
        train_x, valid_x, train_y, valid_y = train_test_split(df['text'], df['stars'])
```

```
In [4]: # TRAIN THE MODEL AND CALCULATE PERFORMANCE METRICS (ACCURACY, PRECISION, RECALL, F-SC
        # FOR BOTH TRAINING AND TEST SET
        # Weighted performance metrics
        def train_model(classifier, feature_vector_train, label, feature_vector_valid):
            # fit the training dataset on the classifier
            classifier.fit(feature_vector_train, label)

            # predict the labels on training dataset (to compare performance metrics against t
            train_predictions = classifier.predict(feature_vector_train)
```

```
    # predict the labels on test dataset
    test_predictions = classifier.predict(feature_vector_valid)

    # metrics for training dataset
    train_accuracy = metrics.accuracy_score(label, train_predictions)
    train_precision = metrics.precision_score(label, train_predictions, average = 'wei
    train_recall = metrics.recall_score(label, train_predictions, average = 'weighted'
    train_f1_score = metrics.f1_score(label, train_predictions, average = 'weighted')

    # metrics for test dataset
    test_accuracy = metrics.accuracy_score(valid_y, test_predictions)
    test_precision = metrics.precision_score(valid_y, test_predictions, average = 'wei
    test_recall = metrics.recall_score(valid_y, test_predictions, average = 'weighted'
    test_f1_score = metrics.f1_score(valid_y, test_predictions, average = 'weighted')

    return [test_accuracy, test_precision, test_recall, test_f1_score], [train_accurac
```

*** Note : the tfidfvectorizer conducts most of the pre-processing steps such as converting to lower case, removing non alpha numeric characters, removing stop words (using max_df). Hence the pre-processing step is not included for logistic regression**

```
In [8]: # word level tf-idf
        tfidf_vect = TfidfVectorizer(lowercase = True, analyzer='word', token_pattern=r'[a-zA-
                                     max_features=500)
        tfidf_vect.fit(df['text'])
        xtrain_tfidf =  tfidf_vect.transform(train_x)
        xvalid_tfidf =  tfidf_vect.transform(valid_x)
```

0.2 Model 1 : Bag of word representation - Word level

```
In [9]: # SVM Classifier on Word Level TF IDF Vectors
        # C (penalty) = 1
        # Gamma = "auto"
        # Kernel : RBF (Default)
        # Default for max_iter is -1 which means there is no limit to the number of iterations
        # First attempt to train the SVM classifier with this default value ran endlessly
        # Second attempt is made by setting max_iter to 1000

        results = train_model(svm.SVC(gamma = "auto", max_iter = 1000), xtrain_tfidf, train_y,

        print ("SVM, WordLevel TF-IDF train accuracy: ", results[1][0])
        print("")
        print ("SVM, WordLevel TF-IDF train precision: ", results[1][1])
        print("")
        print ("SVM, WordLevel TF-IDF train recall: ", results[1][2])
        print("")
        print ("SVM, WordLevel TF-IDF train f1_score: ", results[1][3])
        print("*******************************************************")
        print ("SVM, WordLevel TF-IDF test accuracy: ", results[0][0])
        print("")
        print ("SVM, WordLevel TF-IDF test precision: ", results[0][1])
        print("")
        print ("SVM, WordLevel TF-IDF test recall: ", results[0][2])
        print("")
        print ("SVM, WordLevel TF-IDF test f1_score: ", results[0][3])
```

```
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/svm/base.py:244: ConvergenceWa
  % self.max_iter, ConvergenceWarning)


SVM, WordLevel TF-IDF train accuracy:  0.137424

SVM, WordLevel TF-IDF train precision:  0.21459502128943445

SVM, WordLevel TF-IDF train recall:  0.137424

SVM, WordLevel TF-IDF train f1_score:  0.0622646623396362
*******************************************************
SVM, WordLevel TF-IDF test accuracy:  0.137904

SVM, WordLevel TF-IDF test precision:  0.2274812152626558

SVM, WordLevel TF-IDF test recall:  0.137904

SVM, WordLevel TF-IDF test f1_score:  0.06328901067423066


/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/metrics/classification.py:1143
  'precision', 'predicted', average, warn_for)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/metrics/classification.py:1143
  'precision', 'predicted', average, warn_for)
```

0.3 Model : Bag of word representation - Word level : Change max_iter

```
In [10]: # SVM Classifier on Word Level TF IDF Vectors
         # C (penalty) = 1
         # Gamma = "auto"
         # Kernel : RBF (Default)
         # Third attempt is made by setting max_iter to 10000
         results = train_model(svm.SVC(gamma = "auto", max_iter = 10000), xtrain_tfidf, train_

         print ("SVM, WordLevel TF-IDF train accuracy: ", results[1][0])
         print("")
         print ("SVM, WordLevel TF-IDF train precision: ", results[1][1])
         print("")
         print ("SVM, WordLevel TF-IDF train recall: ", results[1][2])
```

```
        print("")
        print ("SVM, WordLevel TF-IDF train f1_score: ", results[1][3])
        print("*******************************************************")
        print ("SVM, WordLevel TF-IDF test accuracy: ", results[0][0])
        print("")
        print ("SVM, WordLevel TF-IDF test precision: ", results[0][1])
        print("")
        print ("SVM, WordLevel TF-IDF test recall: ", results[0][2])
        print("")
        print ("SVM, WordLevel TF-IDF test f1_score: ", results[0][3])
```

```
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/svm/base.py:244: ConvergenceWa
  % self.max_iter, ConvergenceWarning)


SVM, WordLevel TF-IDF train accuracy:  0.18882666666666667

SVM, WordLevel TF-IDF train precision:  0.26546280430132546

SVM, WordLevel TF-IDF train recall:  0.18882666666666667

SVM, WordLevel TF-IDF train f1_score:  0.17675089850410858
*******************************************************
SVM, WordLevel TF-IDF test accuracy:  0.189192

SVM, WordLevel TF-IDF test precision:  0.26643551590198383

SVM, WordLevel TF-IDF test recall:  0.189192

SVM, WordLevel TF-IDF test f1_score:  0.1774337201487341
```

```
In [5]: # ngram level tf-idf
        tfidf_vect_ngram = TfidfVectorizer(lowercase = True, analyzer='word', token_pattern=r'
                                           ngram_range=(1,3), max_features=500)
        # Fit the model
        tfidf_ngram_transformer = tfidf_vect_ngram.fit(df['text'])
        xtrain_tfidf_ngram =  tfidf_ngram_transformer.transform(train_x)
        xvalid_tfidf_ngram =  tfidf_ngram_transformer.transform(valid_x)

        # Dump the file
        pickle.dump(tfidf_ngram_transformer, open("tfidf_ngram_transformer.pkl", "wb"))
```

0.4 Model 2 : Bag of word representation - Ngram level 1-3 grams

```
In [7]: # SVM Classifier on Ngram Level TF IDF Vectors
        # C (penalty) = 1
        # Gamma = "auto"
        # Kernel : RBF (Default)
        # First attempt to train the SVM classifier with max_iter = 10000 (since it performed 
        # but it ran endlessly for Ngram level
        # Second attempt is made by reducing max_iter to 1000
        results_ngram = train_model(svm.SVC(gamma = "auto", max_iter = 1000), xtrain_tfidf_ngra
        print ("SVM, N-Gram Vectors TF-IDF train accuracy: ", results_ngram[1][0])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF train precision: ", results_ngram[1][1])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF train recall: ", results_ngram[1][2])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF train f1_score: ", results_ngram[1][3])
        print("***************************************************************")
        print ("SVM, N-Gram Vectors TF-IDF test accuracy: ", results_ngram[0][0])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF test precision: ", results_ngram[0][1])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF test recall: ", results_ngram[0][2])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF test f1_score: ", results_ngram[0][3])
```

```
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/svm/base.py:244: ConvergenceWa
  % self.max_iter, ConvergenceWarning)
```

```
SVM, N-Gram Vectors TF-IDF train accuracy:  0.398832

SVM, N-Gram Vectors TF-IDF train precision:  0.32221193737795345

SVM, N-Gram Vectors TF-IDF train recall:  0.398832

SVM, N-Gram Vectors TF-IDF train f1_score:  0.31516400603678385
***************************************************************
SVM, N-Gram Vectors TF-IDF test accuracy:  0.398024

SVM, N-Gram Vectors TF-IDF test precision:  0.32029641839471057

SVM, N-Gram Vectors TF-IDF test recall:  0.398024

SVM, N-Gram Vectors TF-IDF test f1_score:  0.3138662166797743
```

0.5 Model 3 : Bag of word representation - Ngram level 1-3 grams : Change gamma to 1

```
In [9]: # SVM Classifier on Ngram Level TF IDF Vectors
        # C (penalty) = 1
        # Gamma = 1
        # Kernel : RBF (Default)
        # max_iter = 1000
        results_ngram_gamma = train_model(svm.SVC(gamma = 1, max_iter = 1000), xtrain_tfidf_ng
        print ("SVM, N-Gram Vectors TF-IDF train accuracy: ", results_ngram_gamma[1][0])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF train precision: ", results_ngram_gamma[1][1])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF train recall: ", results_ngram_gamma[1][2])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF train f1_score: ", results_ngram_gamma[1][3])
        print("***************************************************************")
        print ("SVM, N-Gram Vectors TF-IDF test accuracy: ", results_ngram_gamma[0][0])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF test precision: ", results_ngram_gamma[0][1])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF test recall: ", results_ngram_gamma[0][2])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF test f1_score: ", results_ngram_gamma[0][3])
```

```
SVM, N-Gram Vectors TF-IDF train accuracy:  0.23055466666666666

SVM, N-Gram Vectors TF-IDF train precision:  0.3166692114832456

SVM, N-Gram Vectors TF-IDF train recall:  0.23055466666666666

SVM, N-Gram Vectors TF-IDF train f1_score:  0.237200153944769
***************************************************************
SVM, N-Gram Vectors TF-IDF test accuracy:  0.227328

SVM, N-Gram Vectors TF-IDF test precision:  0.31015521176665445

SVM, N-Gram Vectors TF-IDF test recall:  0.227328

SVM, N-Gram Vectors TF-IDF test f1_score:  0.23294178086559933
```

0.6 Model 4 : Bag of word representation - Ngram level 1-3 grams, (Keep gamma, other params same as model 2) : Change kernel to linear

```
In [8]: # SVM Classifier on Ngram Level TF IDF Vectors
        # C (penalty) = 1
        # Gamma = "auto"
        # Kernel = Linear
        # Max_iter = 1000

        results_ngram_new = train_model(svm.SVC(kernel = "linear", gamma = "auto", max_iter =
        print ("SVM, N-Gram Vectors TF-IDF train accuracy: ", results_ngram_new[1][0])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF train precision: ", results_ngram_new[1][1])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF train recall: ", results_ngram_new[1][2])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF train f1_score: ", results_ngram_new[1][3])
        print("***************************************************************")
        print ("SVM, N-Gram Vectors TF-IDF test accuracy: ", results_ngram_new[0][0])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF test precision: ", results_ngram_new[0][1])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF test recall: ", results_ngram_new[0][2])
        print("")
        print ("SVM, N-Gram Vectors TF-IDF test f1_score: ", results_ngram_new[0][3])
```

```
SVM, N-Gram Vectors TF-IDF train accuracy:  0.37072533333333335

SVM, N-Gram Vectors TF-IDF train precision:  0.30766569065254

SVM, N-Gram Vectors TF-IDF train recall:  0.37072533333333335

SVM, N-Gram Vectors TF-IDF train f1_score:  0.3140928909583053
***************************************************************
SVM, N-Gram Vectors TF-IDF test accuracy:  0.370408

SVM, N-Gram Vectors TF-IDF test precision:  0.3073791347288203

SVM, N-Gram Vectors TF-IDF test recall:  0.370408

SVM, N-Gram Vectors TF-IDF test f1_score:  0.313741498817925
```

```
In [ ]: # SAVE MODEL SO THAT IT CAN BE LOADED IN THE PREDICT SCRIPT
        best_svm_model = svm.SVC(gamma = "auto", max_iter = 1000)
        best_svm_model.fit(xtrain_tfidf_ngram, train_y)
        filename = 'best_svm_model.sav'
        pickle.dump(best_svm_model, open(filename, 'wb'))
```

```
In [ ]: ## FIT CALIBRATED CLASSIFIER CV TO BE ABLE TO GET PROBABILITIES
        filename = 'best_svm_model.sav'
        loaded_model = pickle.load(open(filename, 'rb'))
        svm_model = CalibratedClassifierCV(loaded_model)
        svm_model.fit(xtrain_tfidf_ngram, train_y)
        pickle.dump(svm_model, open("svm_model.sav", 'wb'))
```

```
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/model_selection/_split.py:205
  warnings.warn(CV_WARNING, FutureWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/svm/base.py:244: ConvergenceWa
  % self.max_iter, ConvergenceWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/svm/base.py:244: ConvergenceWa
  % self.max_iter, ConvergenceWarning)
```

```
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/svm/base.py:244: ConvergenceWa
  % self.max_iter, ConvergenceWarning)
```