SVM on Amazon Food Reviews

May 28, 2018

0.0.1 SVM on Amazon food reviews

In [6]:
```
#importing required Modules
%matplotlib inline
import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import pickle
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import ParameterGrid
from sklearn.svm import SVC
from sklearn.linear_model import SGDClassifier
```

In [7]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [8]:
```
def cleanpunc(sentence):
    '''
    function to clean the word of any punctuation or special characters
    '''
```

```
            cleaned = re.sub(r'[?|!|\'|"|#]',r'',sentence)
            cleaned = re.sub(r'[.|,|)|(|\|/]',r' ',cleaned)
            return  cleaned
        def cleanhtml(sentence):
            '''
            function to clean the word of any html-tags
            '''
            cleanr = re.compile('<.*?>')
            cleantext = re.sub(cleanr, ' ', sentence)
            return cleantext
        def reduce_lengthening(text):
            pattern = re.compile(r"(.)\1{2,}")
            return pattern.sub(r"\1\1", text)

In [9]: #getting stop words
        from nltk.corpus import stopwords
        stop = set(stopwords.words('english'))
        stop.remove('not')
        stop.remove('very')
        #from autocorrect import spell

In [10]: conn = sqlite3.connect('final_clean_LR.sqlite')
         final_review = pd.read_sql_query("""
         SELECT *
         FROM Reviews_final
         """, conn)

In [11]: s = final_review.sample(n=30000,random_state=0)

In [12]: #SORT by time for TBS
         s = s.sort_values(by='Time')

In [13]: s.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 30000 entries, 431 to 339792
Data columns (total 15 columns):
level_0                    30000 non-null int64
index                      30000 non-null int64
Id                         30000 non-null int64
ProductId                  30000 non-null object
UserId                     30000 non-null object
ProfileName                30000 non-null object
HelpfulnessNumerator       30000 non-null int64
HelpfulnessDenominator     30000 non-null int64
Score                      30000 non-null object
Time                       30000 non-null int64
Summary                    30000 non-null object
Text                       30000 non-null object
```

```
CleanedTextBow            30000 non-null object
final_text                30000 non-null object
final_stem_text           30000 non-null object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

```
In [14]: #changing lables to 1 or 0
         s.Score = final_review.Score.apply(lambda x:
                                1 if x == 'positive' else 0)
```

```
In [15]: #Converting to int8
         s.HelpfulnessNumerator = s.HelpfulnessNumerator.astype(np.int8)
         s.HelpfulnessDenominator = s.HelpfulnessDenominator.astype(np.int8)
```

```
In [16]: #Splitting Dataframe for train and test
         train_df = s.iloc[:round(s.shape[0]*0.70),:]
         test_df = s.iloc[round(s.shape[0]*0.70):,:]
```

```
In [17]: train_df.to_csv('train_df_svm.csv',index=False)
         test_df.to_csv('test_df_svm.csv',index=False)
```

```
In [18]: print(train_df.shape)
         print(test_df.shape)
```

```
(21000, 15)
(9000, 15)
```

0.0.2 Bag of Words:

```
In [25]: #BoW with cleaned data and without stopwords
         #simple cv for train data
         scores_train = []
         from nltk.corpus import stopwords
         stop = set(stopwords.words('english'))
         stop.remove('not')
         stop.remove('very')
         #CountVectorizer for BoW
         count_vect = CountVectorizer(stop_words=list(stop),dtype=np.int8)
         X_train = train_df.iloc[:round(train_df.shape[0]*0.70),:]
         X_test_cv = train_df.iloc[round(train_df.shape[0]*0.70):,:]
         final_counts_train = count_vect.fit_transform(
                 X_train['final_text'].values)
         #test
         X_test = count_vect.transform(X_test_cv['final_text'].values)

         scale =StandardScaler(with_mean=False)
         X_train_scale = scale.fit_transform(final_counts_train)
         X_test = scale.transform(X_test)
```

```
In [16]: for i in ParameterGrid({'C':[0.001, 0.01, 0.1, 1, 10],
                                 'gamma':[0.01, 0.001, 0.1, 1]}):
             model = SVC(C=i['C'],kernel='rbf',gamma=i['gamma'])
             model.fit(X_train_scale,X_train.Score)
             train_score = model.score(X_train_scale,X_train.Score)
             test_score = model.score(X_test,X_test_cv.Score)
             print('C',i['C'],'Gamma',i['gamma'],'Train Score',train_score,
                              'Test Score',test_score)
```

```
C 0.001 Gamma 0.01 Train Score 0.8605306122448979 Test Score 0.8275238095238096
C 0.001 Gamma 0.001 Train Score 0.8605306122448979 Test Score 0.8275238095238096
C 0.001 Gamma 0.1 Train Score 0.8605306122448979 Test Score 0.8275238095238096
C 0.001 Gamma 1 Train Score 0.8605306122448979 Test Score 0.8275238095238096
C 0.01 Gamma 0.01 Train Score 0.8605306122448979 Test Score 0.8275238095238096
C 0.01 Gamma 0.001 Train Score 0.8605306122448979 Test Score 0.8275238095238096
C 0.01 Gamma 0.1 Train Score 0.8605306122448979 Test Score 0.8275238095238096
C 0.01 Gamma 1 Train Score 0.8605306122448979 Test Score 0.8275238095238096
C 0.1 Gamma 0.01 Train Score 0.8605306122448979 Test Score 0.8275238095238096
C 0.1 Gamma 0.001 Train Score 0.8605306122448979 Test Score 0.8275238095238096
C 0.1 Gamma 0.1 Train Score 0.8605306122448979 Test Score 0.8275238095238096
C 0.1 Gamma 1 Train Score 0.8605306122448979 Test Score 0.8275238095238096
C 1 Gamma 0.01 Train Score 0.9997959183673469 Test Score 0.8253968253968254
C 1 Gamma 0.001 Train Score 0.9951700680272109 Test Score 0.8274603174603175
C 1 Gamma 0.1 Train Score 1.0 Test Score 0.8253968253968254
C 1 Gamma 1 Train Score 1.0 Test Score 0.8253968253968254
C 10 Gamma 0.01 Train Score 1.0 Test Score 0.8253968253968254
C 10 Gamma 0.001 Train Score 1.0 Test Score 0.8312698412698413
C 10 Gamma 0.1 Train Score 1.0 Test Score 0.8253968253968254
C 10 Gamma 1 Train Score 1.0 Test Score 0.8253968253968254
```

```
In [17]: for i in ParameterGrid({'C':[0.2,0.5,0.8],
                                 'gamma':[0.01, 0.001, 0.0001, 0.005]}):
             model = SVC(C=i['C'],kernel='rbf',gamma=i['gamma'])
             model.fit(X_train_scale,X_train.Score)
             train_score = model.score(X_train_scale,X_train.Score)
             test_score = model.score(X_test,X_test_cv.Score)
             print('C',i['C'],'Gamma',i['gamma'],'Train Score',train_score,
                              'Test Score',test_score)
```

```
C 0.2 Gamma 0.01 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.2 Gamma 0.001 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.2 Gamma 0.0001 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.2 Gamma 0.005 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.5 Gamma 0.01 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.5 Gamma 0.001 Train Score 0.8568707482993198 Test Score 0.8253968253968254
C 0.5 Gamma 0.0001 Train Score 0.8601360544217687 Test Score 0.8271428571428572
C 0.5 Gamma 0.005 Train Score 0.8563945578231292 Test Score 0.8253968253968254
```

```
C 0.8 Gamma 0.01 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.8 Gamma 0.001 Train Score 0.8745578231292517 Test Score 0.8265079365079365
C 0.8 Gamma 0.0001 Train Score 0.9240136054421769 Test Score 0.8353968253968254
C 0.8 Gamma 0.005 Train Score 0.8566666666666667 Test Score 0.8253968253968254
```

Observed that for high C- training data is overfitting so much. for low c. and low values of gamma is giving somewat better scores than high.

```
In [79]: c = [0.005,0.01,0.4,0.8,1.2]
         gamma = [0.000009,0.0008,0.001,0.04,0.2,5,10,15]
         model_grid_bow = GridSearchCV(make_pipeline(CountVectorizer(stop_words=list(stop)),
                                                     StandardScaler(with_mean=False),SVC()),
                                       param_grid={'svc__C': c,'svc__gamma':gamma},
                                      cv=TimeSeriesSplit(n_splits=10),n_jobs=-1)
         model_grid_bow.fit(train_df.final_text,train_df.Score)
```

```
In [56]: dict_scores = []
         idx = 0
         for i in model_grid_bow.grid_scores_:
             dict_score = []
             dict_score.append(i[0]['svc__gamma'])
             dict_score.append(i[0]['svc__C'])
             dict_score.append(i[1])
             dict_score.append(i[2].std())
             dict_score.append(model_grid_bow.cv_results_['mean_train_score'][idx])
             dict_scores.append(dict_score)
             idx = idx + 1
         scores_df = pd.DataFrame(dict_scores,columns=['gamma','C','Test_score',
                                                       'Test_std','Train_score'])
```

```
In [70]: #top scores wit grid search
         scores_df.sort_values('Test_score',ascending=False).head(15)
```

Out[70]:

	gamma	C	Test_score	Test_std	Train_score
33	0.000800	1.200	0.845207	0.019683	0.995821
34	0.001000	1.200	0.844159	0.020326	0.996686
32	0.000009	1.200	0.843478	0.020834	0.883865
25	0.000800	0.800	0.843269	0.021011	0.883656
26	0.001000	0.800	0.842902	0.021275	0.880644
17	0.000800	0.400	0.842378	0.021622	0.869664
18	0.001000	0.400	0.842326	0.021650	0.869558
0	0.000009	0.005	0.842273	0.021690	0.869488
28	0.200000	0.800	0.842273	0.021690	0.869498
23	15.000000	0.400	0.842273	0.021690	0.869488
24	0.000009	0.800	0.842273	0.021703	0.871792
27	0.040000	0.800	0.842273	0.021690	0.869498
30	10.000000	0.800	0.842273	0.021690	0.869498
29	5.000000	0.800	0.842273	0.021690	0.869498
21	5.000000	0.400	0.842273	0.021690	0.869488

```
In [63]: #RandomSearch
         model_random_bow = RandomizedSearchCV(
                             make_pipeline(CountVectorizer(stop_words=list(stop)),
                             StandardScaler(with_mean=False),SVC()),
                             param_distributions={'svc__C': uniform(loc=0,scale=0.7),
                                 'svc__gamma':uniform(loc=0,scale=0.01)},n_iter=15,
                                    cv=TimeSeriesSplit(n_splits=10),n_jobs=-1)
         model_random_bow.fit(train_df.final_text,train_df.Score)

In [66]: dict_scores = []
         idx = 0
         for i in model_random_bow.grid_scores_:
             dict_score = []
             dict_score.append(i[0]['svc__gamma'])
             dict_score.append(i[0]['svc__C'])
             dict_score.append(i[1])
             dict_score.append(i[2].std())
             dict_score.append(model_random_bow.cv_results_['mean_train_score'][idx])
             dict_scores.append(dict_score)
             idx = idx + 1
         scores_df1 = pd.DataFrame(dict_scores,columns=['gamma','C','Test_score',
                                                        'Test_std','Train_score'])

In [72]: scores_df1.sort_values('Test_score',ascending=False).head(10)
```

Out[72]:

	gamma	C	Test_score	Test_std	Train_score
4	0.000415	0.473474	0.842850	0.021271	0.870966
14	0.001121	0.545602	0.842378	0.021622	0.870091
0	0.006791	0.237241	0.842273	0.021690	0.869488
1	0.004997	0.257014	0.842273	0.021690	0.869488
2	0.005213	0.182690	0.842273	0.021690	0.869488
3	0.007353	0.527309	0.842273	0.021690	0.869498
5	0.003778	0.325648	0.842273	0.021690	0.869488
6	0.001744	0.498052	0.842273	0.021690	0.869552
7	0.003124	0.151355	0.842273	0.021690	0.869488
8	0.006712	0.141517	0.842273	0.021690	0.869488

It seems like for high C values, it is giving somewhat better cv score but it is overfitting so much. There is a difference of >15% in train and test scores. so found that `gamma = 0.000800 C = 0.400` are the better params with cv score of `0.842378`

```
In [14]: #BoW with cleaned data and without stopwords and binary
         #simple cv for train data
         scores_train = []
         from nltk.corpus import stopwords
         stop = set(stopwords.words('english'))
         stop.remove('not')
         stop.remove('very')
         #CountVectorizer for BoW
```

```
count_vect = CountVectorizer(stop_words=list(stop),binary=True,dtype=np.int8)
X_train = train_df.iloc[:round(train_df.shape[0]*0.70),:]
X_test_cv = train_df.iloc[round(train_df.shape[0]*0.70):,:]
final_counts_train = count_vect.fit_transform(
        X_train['final_text'].values)
#test
X_test = count_vect.transform(X_test_cv['final_text'].values)
```

```
In [53]: for i in ParameterGrid({'C':[0.4, 0.8, 0.1, 1, 10],
                              'gamma':[0.0008,0.005, 0.1, 1]}):
             model = SVC(C=i['C'],kernel='rbf',gamma=i['gamma'])
             model.fit(final_counts_train,X_train.Score)
             train_score = model.score(final_counts_train,X_train.Score)
             test_score = model.score(X_test,X_test_cv.Score)
             print('C',i['C'],'Gamma',i['gamma'],'Train Score',train_score,
                               'Test Score',test_score)
```

```
C 0.4 Gamma 0.0008 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.4 Gamma 0.005 Train Score 0.8725850340136054 Test Score 0.840952380952381
C 0.4 Gamma 0.1 Train Score 0.8656462585034014 Test Score 0.8284126984126984
C 0.4 Gamma 1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.8 Gamma 0.0008 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.8 Gamma 0.005 Train Score 0.9155102040816326 Test Score 0.8819047619047619
C 0.8 Gamma 0.1 Train Score 0.9710884353741497 Test Score 0.8444444444444444
C 0.8 Gamma 1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 0.0008 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 0.005 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 0.1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 1 Gamma 0.0008 Train Score 0.8570748299319728 Test Score 0.8257142857142857
C 1 Gamma 0.005 Train Score 0.9269387755102041 Test Score 0.8920634920634921
C 1 Gamma 0.1 Train Score 0.9973469387755102 Test Score 0.8485714285714285
C 1 Gamma 1 Train Score 1.0 Test Score 0.8253968253968254
C 10 Gamma 0.0008 Train Score 0.9442857142857143 Test Score 0.9082539682539682
C 10 Gamma 0.005 Train Score 0.9850340136054422 Test Score 0.92
C 10 Gamma 0.1 Train Score 1.0 Test Score 0.8571428571428571
C 10 Gamma 1 Train Score 1.0 Test Score 0.8253968253968254
```

Tried with binary Count vectorizer and found some interesting results with high accuracy for gamma in range of 0.001-0.01 and and c > 0.8, i am getting some high test scores for the data.

```
In [83]: c = [0.8,0.9,1,1.5,3,5,7,10]
         gamma = [0.0008,0.001,0.003,0.005,0.008,0.01,0.05,0.08]
         model_grid_bow_binary = GridSearchCV(make_pipeline(CountVectorizer(stop_words=list(st
                                                            SVC()),
                                              param_grid={'svc__C': c,'svc__gamma':gamma},
                                             cv=TimeSeriesSplit(n_splits=10),n_jobs=-1)
         model_grid_bow_binary.fit(train_df.final_text,train_df.Score)
```

```
In [85]: dict_scores = []
         idx = 0
         for i in model_grid_bow_binary.grid_scores_:
             dict_score = []
             dict_score.append(i[0]['svc__gamma'])
             dict_score.append(i[0]['svc__C'])
             dict_score.append(i[1])
             dict_score.append(i[2].std())
             dict_score.append(model_grid_bow_binary.cv_results_['mean_train_score'][idx])
             dict_scores.append(dict_score)
             idx = idx + 1
         scores_df_bin = pd.DataFrame(dict_scores,columns=['gamma','C','Test_score',
                                                           'Test_std','Train_score'])

In [87]: #top scores
         scores_df_bin.sort_values('Test_score',ascending=False).head(10)
```

Out[87]:

	gamma	C	Test_score	Test_std	Train_score
53	0.010	7.0	0.921215	0.005992	0.993688
61	0.010	10.0	0.920639	0.005565	0.996447
45	0.010	5.0	0.920377	0.006399	0.989241
60	0.008	10.0	0.920063	0.005074	0.994219
52	0.008	7.0	0.920010	0.005494	0.990022
59	0.005	10.0	0.919329	0.005046	0.986794
44	0.008	5.0	0.919068	0.006339	0.984562
51	0.005	7.0	0.918649	0.007152	0.980237
58	0.003	10.0	0.917444	0.007297	0.975592
37	0.010	3.0	0.916763	0.009404	0.979108

```
In [99]: #random search
         model_random_bow_binary = RandomizedSearchCV(
                             make_pipeline(CountVectorizer(stop_words=list(stop),binary=True),
                                     SVC()),
                           param_distributions={'svc__C': uniform(loc=0,scale=10),
                          'svc__gamma':uniform(loc=0.003,scale=0.017)},n_iter=20,
                                     cv=TimeSeriesSplit(n_splits=10),n_jobs=-1)
         model_random_bow_binary.fit(train_df.final_text,train_df.Score)

In [100]: dict_scores = []
          idx = 0
          for i in model_random_bow_binary.grid_scores_:
              dict_score = []
              dict_score.append(i[0]['svc__gamma'])
              dict_score.append(i[0]['svc__C'])
              dict_score.append(i[1])
              dict_score.append(i[2].std())
              dict_score.append(model_random_bow_binary.cv_results_['mean_train_score'][idx])
              dict_scores.append(dict_score)
              idx = idx + 1
```

```
scores_df1_bin = pd.DataFrame(dict_scores,columns=['gamma','C','Test_score',
                                                   'Test_std','Train_score'])
```

```
In [103]: #top scores
          scores_df1_bin.sort_values('Test_score',ascending=False).head(10)
```

Out[103]:

	gamma	C	Test_score	Test_std	Train_score
9	0.012632	5.387201	0.921320	0.006144	0.994085
7	0.010988	8.434631	0.920744	0.005843	0.996239
18	0.013324	5.211789	0.920691	0.006230	0.994289
13	0.015818	4.657298	0.920482	0.007147	0.995270
0	0.008742	6.182782	0.920377	0.005578	0.989817
6	0.015201	7.367153	0.920272	0.007147	0.997701
2	0.017782	4.186368	0.920168	0.007176	0.995562
14	0.006675	7.241270	0.919906	0.006305	0.987144
1	0.018738	5.846448	0.919749	0.007964	0.997884
15	0.015909	8.272383	0.919644	0.008027	0.998387

Comapared to Non-binary Bag of word binary bag of words score was high and best score found at `gamma = 0.010 ,C = 7.0` and cv mean score is 0.921215

```
In [21]: #test scores
         scores_train = []
         from nltk.corpus import stopwords
         stop = set(stopwords.words('english'))
         stop.remove('not')
         stop.remove('very')
         #CountVectorizer for BoW
         count_vect = CountVectorizer(stop_words=list(stop),binary=True,dtype=np.int8)
         final_counts_train = count_vect.fit_transform(
                 train_df['final_text'].values)
         #test
         X_test = count_vect.transform(test_df['final_text'].values)

         model = SVC(C=7,kernel='rbf',gamma=0.010)
         model.fit(final_counts_train,train_df.Score)
         #Predicting training data
         train_list = model.predict(final_counts_train)
         #Accuracy score
         score_train = accuracy_score(train_df.Score,train_list)
         #predict test cv
         test_list = model.predict(X_test)
         #Accuracy score
         score_test = accuracy_score(test_df.Score,test_list)
         #precision
         #precision
         test_precision = precision_score(test_df.Score,test_list)
         #recall
         test_recall = recall_score(test_df.Score,test_list)
```

```
        #confusion matrix
        confusion_matrix_test = confusion_matrix(test_df.Score,test_list)
        print('C' ,7,'gamma',0.010)
        print('Train Score', score_train)
        print('Test Score',score_test)
        print('Test Precision',test_precision)
        print('Test Recall',test_recall)
        print('Test ConfusionMatrix',confusion_matrix_test)
```

```
C 7 gamma 0.01
Train Score 0.9915238095238095
Test Score 0.9246666666666666
Test Precision 0.9447222953408791
Test Recall 0.9653039268423884
Test ConfusionMatrix [[1144  420]
 [ 258 7178]]
```

```
In [22]: #no of support vectrors for each class
         model.n_support_
```

```
Out[22]: array([2245, 3591], dtype=int32)
```

SGD With BoW

```
In [88]: #random search
         model_random_bow_binary = RandomizedSearchCV(make_pipeline(
                                        CountVectorizer(stop_words=list(stop),binary=True),
                                         SGDClassifier(n_jobs=-1)),
                                param_distributions={'sgdclassifier__penalty':['l1','l2'],
                                   'sgdclassifier__alpha':uniform(loc=0.00001,scale=0.069),
                                   'sgdclassifier__l1_ratio':uniform(loc=0,scale=1)},
                                                        n_iter=100,
                                    cv=TimeSeriesSplit(n_splits=10),n_jobs=-1)
         model_random_bow_binary.fit(train_df.final_text,train_df.Score)
```

```
In [89]: dict_scores = []
         idx = 0
         for i in model_random_bow_binary.grid_scores_:
             dict_score = []
             dict_score.append(i[0]['sgdclassifier__alpha'])
             dict_score.append(i[0]['sgdclassifier__l1_ratio'])
             dict_score.append(i[0]['sgdclassifier__penalty'])
             dict_score.append(i[1])
             dict_score.append(i[2].std())
             dict_score.append(model_random_bow_binary.cv_results_['mean_train_score'][idx])
             dict_scores.append(dict_score)
             idx = idx + 1
         scores_df1_bin = pd.DataFrame(dict_scores,columns=['alpha','l1_rato','penality','Test_
                                                            'Test_std','Train_score'])
```

```
In [90]: scores_df1_bin.sort_values('Test_score',ascending=False).head(10)
```

Out[90]:

	alpha	l1_rato	penality	Test_score	Test_std	Train_score
13	0.003570	0.912537	l2	0.915977	0.007592	0.961918
95	0.000654	0.544642	l2	0.915872	0.006177	0.980992
38	0.000256	0.797285	l2	0.910529	0.006600	0.983756
46	0.007463	0.314634	l2	0.906129	0.010484	0.946527
80	0.010533	0.760320	l2	0.901414	0.010897	0.937890
7	0.000199	0.744540	l1	0.900681	0.006887	0.962679
63	0.011535	0.273312	l2	0.900210	0.011199	0.935924
56	0.012332	0.480412	l2	0.898638	0.010939	0.934297
22	0.013868	0.611535	l2	0.893557	0.009607	0.928721
77	0.015297	0.045345	l2	0.892981	0.011083	0.926446

Got best mean cv at alpha = 0.003570. l1_ratio = 0.912537 and penality l2 and corresponding mean cv test score is 0.915977

```
In [93]: #test scores
         scores_train = []
         from nltk.corpus import stopwords
         stop = set(stopwords.words('english'))
         stop.remove('not')
         stop.remove('very')
         #CountVectorizer for BoW
         count_vect = CountVectorizer(stop_words=list(stop),binary=True,dtype=np.int8)
         final_counts_train = count_vect.fit_transform(
                 train_df['final_text'].values)
         #test
         X_test = count_vect.transform(test_df['final_text'].values)

         model = SGDClassifier(penalty='l2',alpha=0.003570,l1_ratio=0.912537,n_jobs=-1) #0.003
         model.fit(final_counts_train,train_df.Score)
         #Predicting training data
         train_list = model.predict(final_counts_train)
         #Accuracy score
         score_train = accuracy_score(train_df.Score,train_list)
         #predict test cv
         test_list = model.predict(X_test)
         #Accuracy score
         score_test = accuracy_score(test_df.Score,test_list)
         #precision
         #precision
         test_precision = precision_score(test_df.Score,test_list)
         #recall
         test_recall = recall_score(test_df.Score,test_list)
         #confusion matrix
         confusion_matrix_test = confusion_matrix(test_df.Score,test_list)
         print("With SGD Classifier penalty='l2',alpha=0.003570,l1_ratio=0.912537 ")
```

```
        print('Train Score', score_train)
        print('Test Score',score_test)
        print('Test Precision',test_precision)
        print('Test Recall',test_recall)
        print('Test ConfusionMatrix',confusion_matrix_test)
```

```
With SGD Classifier penalty='l2',alpha=0.003570,l1_ratio=0.912537
Train Score 0.9445238095238095
Test Score 0.9197777777777778
Test Precision 0.930053804765565
Test Recall 0.9763313609467456
Test ConfusionMatrix [[1018  546]
 [ 176 7260]]
```

0.0.3 Tf-Idf

```
In [13]: #TFIDF with (1,2) gram with cleaned data
         #simple cv for train data
         #tfidf vec
         tf_idf_vect = TfidfVectorizer(ngram_range=(1,2))
         X_train = train_df.iloc[:round(train_df.shape[0]*0.70),:]
         X_test_cv = train_df.iloc[round(train_df.shape[0]*0.70):,:]
         final_counts_train = tf_idf_vect.fit_transform(
                 X_train['final_text'].values)
         #test
         X_test = tf_idf_vect.transform(X_test_cv['final_text'].values)
```

```
In [24]: for i in ParameterGrid({'C':[0.001,0.01,0.1,1,5,10],
                                 'gamma':[0.001,0.008,0.01,0.1,0.5,1,10]}):
             model = SVC(C=i['C'],kernel='rbf',gamma=i['gamma'])
             model.fit(final_counts_train,X_train.Score)
             train_score = model.score(final_counts_train,X_train.Score)
             test_score = model.score(X_test,X_test_cv.Score)
             print('C',i['C'],'Gamma',i['gamma'],'Train Score',train_score,
                              'Test Score',test_score)
```

```
C 0.001 Gamma 0.001 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 0.008 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 0.01 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 0.1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 0.5 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 10 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 0.001 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 0.008 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 0.01 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 0.1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 0.5 Train Score 0.8563945578231292 Test Score 0.8253968253968254
```

```
C 0.01 Gamma 1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 10 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 0.001 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 0.008 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 0.01 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 0.1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 0.5 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 10 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 1 Gamma 0.001 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 1 Gamma 0.008 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 1 Gamma 0.01 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 1 Gamma 0.1 Train Score 0.8623129251700681 Test Score 0.8311111111111111
C 1 Gamma 0.5 Train Score 0.978843537414966 Test Score 0.9011111111111111
C 1 Gamma 1 Train Score 0.9975510204081632 Test Score 0.8942857142857142
C 1 Gamma 10 Train Score 1.0 Test Score 0.8253968253968254
C 5 Gamma 0.001 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 5 Gamma 0.008 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 5 Gamma 0.01 Train Score 0.8564625850340136 Test Score 0.8253968253968254
C 5 Gamma 0.1 Train Score 0.9954421768707483 Test Score 0.9288888888888889
C 5 Gamma 0.5 Train Score 1.0 Test Score 0.93
C 5 Gamma 1 Train Score 1.0 Test Score 0.9146031746031746
C 5 Gamma 10 Train Score 1.0 Test Score 0.8253968253968254
C 10 Gamma 0.001 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 10 Gamma 0.008 Train Score 0.8604081632653061 Test Score 0.8293650793650794
C 10 Gamma 0.01 Train Score 0.8687074829931973 Test Score 0.8380952380952381
C 10 Gamma 0.1 Train Score 0.9999319727891156 Test Score 0.9382539682539682
C 10 Gamma 0.5 Train Score 1.0 Test Score 0.93
C 10 Gamma 1 Train Score 1.0 Test Score 0.9146031746031746
C 10 Gamma 10 Train Score 1.0 Test Score 0.8253968253968254
```

```
In [66]: c = [0.1,0.5,0.8,1,5,7,10,20]
         gamma = [0.008,0.007,0.1,0.3,0.5,1,3,10]
         model_grid_tfidf = GridSearchCV(make_pipeline(TfidfVectorizer(ngram_range=(1,2)),
                                                       SVC()),
                                         param_grid={'svc__C': c,'svc__gamma':gamma},
                                        cv=TimeSeriesSplit(n_splits=10),n_jobs=-1)
         model_grid_tfidf.fit(train_df.final_text,train_df.Score)
```

```
In [67]: dict_scores = []
         idx = 0
         for i in model_grid_tfidf.grid_scores_:
             dict_score = []
             dict_score.append(i[0]['svc__gamma'])
             dict_score.append(i[0]['svc__C'])
             dict_score.append(i[1])
             dict_score.append(i[2].std())
```

```
        dict_score.append(model_grid_tfidf.cv_results_['mean_train_score'][idx])
        dict_scores.append(dict_score)
        idx = idx + 1
    scores_df = pd.DataFrame(dict_scores,columns=['gamma','C','Test_score',
                                                  'Test_std','Train_score'])
```

```
In [71]: scores_df.sort_values('Test_score',ascending=False).head(10)
```

Out[71]:

	gamma	C	Test_score	Test_std	Train_score
50	0.1	10.0	0.925982	0.013537	0.999961
58	0.1	20.0	0.925930	0.013911	1.000000
42	0.1	7.0	0.922263	0.014279	0.999523
35	0.3	5.0	0.922263	0.014587	1.000000
51	0.3	10.0	0.922211	0.014668	1.000000
43	0.3	7.0	0.922211	0.014668	1.000000
59	0.3	20.0	0.922211	0.014668	1.000000
36	0.5	5.0	0.918806	0.014783	1.000000
52	0.5	10.0	0.918806	0.014783	1.000000
44	0.5	7.0	0.918806	0.014783	1.000000

for high values of c model is overfitting to train data and for each c with reasonable gamma is giving good score than low or high gamma.

```
In [12]: model_random_tfidf = RandomizedSearchCV(
                              make_pipeline(TfidfVectorizer(ngram_range=(1,2)),SVC()),
                              param_distributions={'svc__C':uniform(loc=0,scale=12),
                                                   'svc__gamma':uniform(loc=0,scale=0.7)},
                                        n_iter=20,cv=TimeSeriesSplit(n_splits=10),n_jobs=-1)
    model_random_tfidf.fit(train_df.final_text,train_df.Score)
```

```
In [13]: dict_scores = []
    idx = 0
    for i in model_random_tfidf.grid_scores_:
        dict_score = []
        dict_score.append(i[0]['svc__gamma'])
        dict_score.append(i[0]['svc__C'])
        dict_score.append(i[1])
        dict_score.append(i[2].std())
        dict_score.append(model_random_tfidf.cv_results_['mean_train_score'][idx])
        dict_scores.append(dict_score)
        idx = idx + 1
    scores_df1 = pd.DataFrame(dict_scores,columns=['gamma','C','Test_score',
                                                   'Test_std','Train_score'])
```

```
In [17]: #top scores with random search
    scores_df1.sort_values('Test_score',ascending=False).head(10)
```

Out[17]:

	gamma	C	Test_score	Test_std	Train_score
13	0.175194	7.107109	0.924987	0.014313	1.000000

```
4   0.205345   7.555669    0.924620  0.014065     1.000000
0   0.192857   4.893355    0.923258  0.014181     0.999950
8   0.266751   8.332227    0.923206  0.014551     1.000000
11  0.324423   5.865958    0.922001  0.014653     1.000000
15  0.311839  10.320605    0.921896  0.014627     1.000000
18  0.273510   3.244258    0.921320  0.014689     0.999898
19  0.347285   9.050680    0.921163  0.014763     1.000000
6   0.430498  11.934278    0.920430  0.014574     1.000000
14  0.068950   8.277344    0.919958  0.015449     0.998265
```

From 10 fold cv got high mean cv at `gamma = 0.175194, C = 7.107109` and mean cv is 0.924987

In [19]:
```
#test scores
#TFIDF with (1,2) gram with cleaned data
#tfidf vec
tf_idf_vect = TfidfVectorizer(ngram_range=(1,2))
final_counts_train = tf_idf_vect.fit_transform(
        train_df['final_text'].values)
#test
X_test = tf_idf_vect.transform(test_df['final_text'].values)

model = SVC(C=7.107109,kernel='rbf',gamma=0.175194)
model.fit(final_counts_train,train_df.Score)
#Predicting training data
train_list = model.predict(final_counts_train)
#Accuracy score
score_train = accuracy_score(train_df.Score,train_list)
#predict test cv
test_list = model.predict(X_test)
#Accuracy score
score_test = accuracy_score(test_df.Score,test_list)
#precision
#precision
test_precision = precision_score(test_df.Score,test_list)
#recall
test_recall = recall_score(test_df.Score,test_list)
#confusion matrix
confusion_matrix_test = confusion_matrix(test_df.Score,test_list)
print('C' ,7,'gamma',0.175194)
print('Train Score', score_train)
print('Test Score',score_test)
print('Test Precision',test_precision)
print('Test Recall',test_recall)
print('Test ConfusionMatrix',confusion_matrix_test)
print('No of support vectors for each class',model.n_support_)
```

```
C 7 gamma 0.175194
Train Score 1.0
```

```
Test Score 0.9415555555555556
Test Precision 0.9508089770354906
Test Recall 0.9799623453469607
Test ConfusionMatrix [[1187  377]
 [ 149 7287]]
No of support vectors for each class [2810 6720]
```

SGD Classifier

```
In [14]: model_random_tfidf = RandomizedSearchCV(make_pipeline(TfidfVectorizer(ngram_range=(1,2
                                                        SGDClassifier(n_jobs=-1)),
                                            param_distributions={'sgdclassifier__penalty':['l1','l2']
                                               'sgdclassifier__alpha':uniform(loc=0.00001,scale=0.069
                                                'sgdclassifier__l1_ratio':uniform(loc=0,scale=1)},n_
                                                     ,cv=TimeSeriesSplit(n_splits=10),n_jobs=-1)
         model_random_tfidf.fit(train_df.final_text,train_df.Score)

In [18]: dict_scores = []
         idx = 0
         for i in model_random_tfidf.grid_scores_:
             dict_score = []
             dict_score.append(i[0]['sgdclassifier__alpha'])
             dict_score.append(i[0]['sgdclassifier__l1_ratio'])
             dict_score.append(i[0]['sgdclassifier__penalty'])
             dict_score.append(i[1])
             dict_score.append(i[2].std())
             dict_score.append(model_random_tfidf.cv_results_['mean_train_score'][idx])
             dict_scores.append(dict_score)
             idx = idx + 1
         scores_df1 = pd.DataFrame(dict_scores,columns=['alpha','l1_ratio','penality','Test_sco
                                                        'Test_std','Train_score'])

In [19]: scores_df1.sort_values('Test_score',ascending=False).head(10)
```

Out[19]:

	alpha	l1_ratio	penality	Test_score	Test_std	Train_score
206	0.000116	0.738118	l2	0.912729	0.007599	0.983543
263	0.000233	0.550213	l2	0.878104	0.017691	0.935960
153	0.000264	0.305034	l2	0.870875	0.019746	0.925270
184	0.000207	0.538170	l1	0.865008	0.015358	0.885320
29	0.000437	0.431934	l2	0.847564	0.023918	0.886908
268	0.000545	0.509713	l2	0.843426	0.022213	0.879669
179	0.000690	0.546427	l2	0.842326	0.021736	0.873685
202	0.043333	0.990595	l1	0.842273	0.021690	0.869488
201	0.025632	0.245308	l1	0.842273	0.021690	0.869488
200	0.016020	0.509449	l1	0.842273	0.021690	0.869488

```
In [24]: for i in ParameterGrid({'alpha':[0.00005,0.00008,0.0001,0.00012,0.00018,0.00023],
                                'l1_ratio':[0,0.03,0.05,0.08,0.1,0.15,0.25,0.35,0.45,0.55,0.65,0.7
```

```
                        'penality':['l1','l2']}):
            model = SGDClassifier(penalty=i['penality'],alpha=i['alpha'],l1_ratio=i['l1_ratio
            model.fit(final_counts_train,X_train.Score)
            train_score = model.score(final_counts_train,X_train.Score)
            test_score = model.score(X_test,X_test_cv.Score)
            print('Alpha',i['alpha'],'l1_ratio',i['l1_ratio'],'Penality',i['penality'],
                  'Train Score',train_score,'Test Score',test_score)
```

```
Alpha 5e-05 l1_ratio 0 Penality l1 Train Score 0.9470748299319728 Test Score 0.930317460317460
Alpha 5e-05 l1_ratio 0 Penality l2 Train Score 0.998639455782313 Test Score 0.9344444444444444
Alpha 5e-05 l1_ratio 0.03 Penality l1 Train Score 0.9455102040816327 Test Score 0.927936507936
Alpha 5e-05 l1_ratio 0.03 Penality l2 Train Score 0.9984353741496599 Test Score 0.935079365079
Alpha 5e-05 l1_ratio 0.05 Penality l1 Train Score 0.9473469387755102 Test Score 0.928095238095
Alpha 5e-05 l1_ratio 0.05 Penality l2 Train Score 0.9985034013605442 Test Score 0.935079365079
Alpha 5e-05 l1_ratio 0.08 Penality l1 Train Score 0.9428571428571428 Test Score 0.924444444444
Alpha 5e-05 l1_ratio 0.08 Penality l2 Train Score 0.998639455782313 Test Score 0.932698412698
Alpha 5e-05 l1_ratio 0.1 Penality l1 Train Score 0.9444897959183673 Test Score 0.928730158730
Alpha 5e-05 l1_ratio 0.1 Penality l2 Train Score 0.998639455782313 Test Score 0.935714285714285
Alpha 5e-05 l1_ratio 0.15 Penality l1 Train Score 0.9457142857142857 Test Score 0.927142857142
Alpha 5e-05 l1_ratio 0.15 Penality l2 Train Score 0.9985714285714286 Test Score 0.933968253968
Alpha 5e-05 l1_ratio 0.25 Penality l1 Train Score 0.9439455782312925 Test Score 0.924603174603
Alpha 5e-05 l1_ratio 0.25 Penality l2 Train Score 0.9985714285714286 Test Score 0.935555555555
Alpha 5e-05 l1_ratio 0.35 Penality l1 Train Score 0.9451020408163265 Test Score 0.925714285714
Alpha 5e-05 l1_ratio 0.35 Penality l2 Train Score 0.9983673469387755 Test Score 0.932857142857
Alpha 5e-05 l1_ratio 0.45 Penality l1 Train Score 0.9470748299319728 Test Score 0.928412698412
Alpha 5e-05 l1_ratio 0.45 Penality l2 Train Score 0.998639455782313 Test Score 0.936825396825
Alpha 5e-05 l1_ratio 0.55 Penality l1 Train Score 0.9471428571428572 Test Score 0.929523809523
Alpha 5e-05 l1_ratio 0.55 Penality l2 Train Score 0.998639455782313 Test Score 0.935079365079
Alpha 5e-05 l1_ratio 0.65 Penality l1 Train Score 0.9452380952380952 Test Score 0.926666666666
Alpha 5e-05 l1_ratio 0.65 Penality l2 Train Score 0.9983673469387755 Test Score 0.932539682539
Alpha 5e-05 l1_ratio 0.75 Penality l1 Train Score 0.9455102040816327 Test Score 0.926507936507
Alpha 5e-05 l1_ratio 0.75 Penality l2 Train Score 0.9984353741496599 Test Score 0.935873015873
Alpha 5e-05 l1_ratio 0.85 Penality l1 Train Score 0.947687074829932 Test Score 0.929841269841
Alpha 5e-05 l1_ratio 0.85 Penality l2 Train Score 0.9985034013605442 Test Score 0.933650793650
Alpha 5e-05 l1_ratio 0.95 Penality l1 Train Score 0.9448979591836735 Test Score 0.925238095238
Alpha 5e-05 l1_ratio 0.95 Penality l2 Train Score 0.9985714285714286 Test Score 0.936984126984
Alpha 8e-05 l1_ratio 0 Penality l1 Train Score 0.9295918367346939 Test Score 0.914761904761904
Alpha 8e-05 l1_ratio 0 Penality l2 Train Score 0.9891836734693877 Test Score 0.926825396825396
Alpha 8e-05 l1_ratio 0.03 Penality l1 Train Score 0.9272789115646258 Test Score 0.914126984126
Alpha 8e-05 l1_ratio 0.03 Penality l2 Train Score 0.9894557823129252 Test Score 0.926984126984
Alpha 8e-05 l1_ratio 0.05 Penality l1 Train Score 0.9251700680272109 Test Score 0.912539682539
Alpha 8e-05 l1_ratio 0.05 Penality l2 Train Score 0.9907482993197279 Test Score 0.928095238095
Alpha 8e-05 l1_ratio 0.08 Penality l1 Train Score 0.9274829931972789 Test Score 0.913015873015
Alpha 8e-05 l1_ratio 0.08 Penality l2 Train Score 0.9907482993197279 Test Score 0.927936507936
Alpha 8e-05 l1_ratio 0.1 Penality l1 Train Score 0.9248979591836735 Test Score 0.913968253968
Alpha 8e-05 l1_ratio 0.1 Penality l2 Train Score 0.988843537414966 Test Score 0.924761904761904
Alpha 8e-05 l1_ratio 0.15 Penality l1 Train Score 0.9279591836734694 Test Score 0.912857142857
Alpha 8e-05 l1_ratio 0.15 Penality l2 Train Score 0.988843537414966 Test Score 0.925555555555
```

```
Alpha 8e-05 l1_ratio 0.25 Penality l1 Train Score 0.9262585034013605 Test Score 0.9096825396825
Alpha 8e-05 l1_ratio 0.25 Penality l2 Train Score 0.9885034013605443 Test Score 0.9266666666666
Alpha 8e-05 l1_ratio 0.35 Penality l1 Train Score 0.9283673469387755 Test Score 0.9153968253968
Alpha 8e-05 l1_ratio 0.35 Penality l2 Train Score 0.9893877551020408 Test Score 0.926031746031
Alpha 8e-05 l1_ratio 0.45 Penality l1 Train Score 0.9261224489795918 Test Score 0.9114285714285
Alpha 8e-05 l1_ratio 0.45 Penality l2 Train Score 0.9889795918367347 Test Score 0.9247619047619
Alpha 8e-05 l1_ratio 0.55 Penality l1 Train Score 0.9268027210884354 Test Score 0.9123809523809
Alpha 8e-05 l1_ratio 0.55 Penality l2 Train Score 0.9903401360544217 Test Score 0.9285714285714
Alpha 8e-05 l1_ratio 0.65 Penality l1 Train Score 0.9248979591836735 Test Score 0.9084126984126
Alpha 8e-05 l1_ratio 0.65 Penality l2 Train Score 0.9908163265306122 Test Score 0.9293650793650
Alpha 8e-05 l1_ratio 0.75 Penality l1 Train Score 0.9285034013605442 Test Score 0.9136507936507
Alpha 8e-05 l1_ratio 0.75 Penality l2 Train Score 0.9907482993197279 Test Score 0.9287301587301
Alpha 8e-05 l1_ratio 0.85 Penality l1 Train Score 0.926734693877551 Test Score 0.9125396825396
Alpha 8e-05 l1_ratio 0.85 Penality l2 Train Score 0.9890476190476191 Test Score 0.9249206349206
Alpha 8e-05 l1_ratio 0.95 Penality l1 Train Score 0.9276190476190476 Test Score 0.911904761904
Alpha 8e-05 l1_ratio 0.95 Penality l2 Train Score 0.991156462585034 Test Score 0.93
Alpha 0.0001 l1_ratio 0 Penality l1 Train Score 0.9206122448979592 Test Score 0.908253968253968
Alpha 0.0001 l1_ratio 0 Penality l2 Train Score 0.976326530612245 Test Score 0.9174603174603174
Alpha 0.0001 l1_ratio 0.03 Penality l1 Train Score 0.9180272108843538 Test Score 0.9038095238095
Alpha 0.0001 l1_ratio 0.03 Penality l2 Train Score 0.9778231292517007 Test Score 0.9179365079365
Alpha 0.0001 l1_ratio 0.05 Penality l1 Train Score 0.9162585034013605 Test Score 0.9028571428571
Alpha 0.0001 l1_ratio 0.05 Penality l2 Train Score 0.9795918367346939 Test Score 0.9193650793650
Alpha 0.0001 l1_ratio 0.08 Penality l1 Train Score 0.9176190476190477 Test Score 0.9042857142857
Alpha 0.0001 l1_ratio 0.08 Penality l2 Train Score 0.9795918367346939 Test Score 0.9206349206349
Alpha 0.0001 l1_ratio 0.1 Penality l1 Train Score 0.9201360544217687 Test Score 0.9066666666666
Alpha 0.0001 l1_ratio 0.1 Penality l2 Train Score 0.9780952380952381 Test Score 0.9193650793650
Alpha 0.0001 l1_ratio 0.15 Penality l1 Train Score 0.9177551020408163 Test Score 0.9052380952380
Alpha 0.0001 l1_ratio 0.15 Penality l2 Train Score 0.9774149659863945 Test Score 0.9184126984126
Alpha 0.0001 l1_ratio 0.25 Penality l1 Train Score 0.916530612244898 Test Score 0.9020634920634
Alpha 0.0001 l1_ratio 0.25 Penality l2 Train Score 0.9759183673469388 Test Score 0.9161904761904
Alpha 0.0001 l1_ratio 0.35 Penality l1 Train Score 0.9171428571428571 Test Score 0.9050793650793
Alpha 0.0001 l1_ratio 0.35 Penality l2 Train Score 0.9795238095238096 Test Score 0.9215873015873
Alpha 0.0001 l1_ratio 0.45 Penality l1 Train Score 0.917687074829932 Test Score 0.9047619047619
Alpha 0.0001 l1_ratio 0.45 Penality l2 Train Score 0.9797959183673469 Test Score 0.9207936507936
Alpha 0.0001 l1_ratio 0.55 Penality l1 Train Score 0.9180952380952381 Test Score 0.9047619047619
Alpha 0.0001 l1_ratio 0.55 Penality l2 Train Score 0.98 Test Score 0.9203174603174603
Alpha 0.0001 l1_ratio 0.65 Penality l1 Train Score 0.9184353741496598 Test Score 0.9055555555555
Alpha 0.0001 l1_ratio 0.65 Penality l2 Train Score 0.9772789115646259 Test Score 0.9169841269841
Alpha 0.0001 l1_ratio 0.75 Penality l1 Train Score 0.9205442176870748 Test Score 0.9071428571428
Alpha 0.0001 l1_ratio 0.75 Penality l2 Train Score 0.9800680272108844 Test Score 0.9220634920634
Alpha 0.0001 l1_ratio 0.85 Penality l1 Train Score 0.9163945578231293 Test Score 0.9012698412698
Alpha 0.0001 l1_ratio 0.85 Penality l2 Train Score 0.9791836734693877 Test Score 0.9214285714285
Alpha 0.0001 l1_ratio 0.95 Penality l1 Train Score 0.9194557823129251 Test Score 0.9061904761904
Alpha 0.0001 l1_ratio 0.95 Penality l2 Train Score 0.9761224489795919 Test Score 0.9180952380952
Alpha 0.00012 l1_ratio 0 Penality l1 Train Score 0.9089795918367347 Test Score 0.8968253968253968
Alpha 0.00012 l1_ratio 0 Penality l2 Train Score 0.9679591836734693 Test Score 0.9117460317460317
Alpha 0.00012 l1_ratio 0.03 Penality l1 Train Score 0.9079591836734694 Test Score 0.8942857142857
Alpha 0.00012 l1_ratio 0.03 Penality l2 Train Score 0.9718367346938775 Test Score 0.9139682539682
```

```
Alpha 0.00012 l1_ratio 0.05 Penality l1 Train Score 0.9091156462585034 Test Score 0.8969841269
Alpha 0.00012 l1_ratio 0.05 Penality l2 Train Score 0.9671428571428572 Test Score 0.9101587301
Alpha 0.00012 l1_ratio 0.08 Penality l1 Train Score 0.9091836734693878 Test Score 0.8949206349
Alpha 0.00012 l1_ratio 0.08 Penality l2 Train Score 0.9682312925170068 Test Score 0.9120634920
Alpha 0.00012 l1_ratio 0.1 Penality l1 Train Score 0.9068707482993197 Test Score 0.894761904761
Alpha 0.00012 l1_ratio 0.1 Penality l2 Train Score 0.9696598639455782 Test Score 0.912380952380
Alpha 0.00012 l1_ratio 0.15 Penality l1 Train Score 0.9097278911564626 Test Score 0.897301587301
Alpha 0.00012 l1_ratio 0.15 Penality l2 Train Score 0.9698639455782313 Test Score 0.9144444444
Alpha 0.00012 l1_ratio 0.25 Penality l1 Train Score 0.9082993197278911 Test Score 0.895079365079
Alpha 0.00012 l1_ratio 0.25 Penality l2 Train Score 0.9674829931972789 Test Score 0.911904761904
Alpha 0.00012 l1_ratio 0.35 Penality l1 Train Score 0.9074829931972789 Test Score 0.8949206349
Alpha 0.00012 l1_ratio 0.35 Penality l2 Train Score 0.969047619047619 Test Score 0.912222222222
Alpha 0.00012 l1_ratio 0.45 Penality l1 Train Score 0.9074829931972789 Test Score 0.894761904761
Alpha 0.00012 l1_ratio 0.45 Penality l2 Train Score 0.9695238095238096 Test Score 0.913174603174
Alpha 0.00012 l1_ratio 0.55 Penality l1 Train Score 0.9112925170068027 Test Score 0.898730158730
Alpha 0.00012 l1_ratio 0.55 Penality l2 Train Score 0.9710884353741497 Test Score 0.912539682539
Alpha 0.00012 l1_ratio 0.65 Penality l1 Train Score 0.9098639455782312 Test Score 0.897936507936
Alpha 0.00012 l1_ratio 0.65 Penality l2 Train Score 0.968843537414966 Test Score 0.911746031746
Alpha 0.00012 l1_ratio 0.75 Penality l1 Train Score 0.9104081632653062 Test Score 0.897777777777
Alpha 0.00012 l1_ratio 0.75 Penality l2 Train Score 0.9705442176870749 Test Score 0.9144444444
Alpha 0.00012 l1_ratio 0.85 Penality l1 Train Score 0.9074149659863946 Test Score 0.895079365079
Alpha 0.00012 l1_ratio 0.85 Penality l2 Train Score 0.9674149659863945 Test Score 0.911111111111
Alpha 0.00012 l1_ratio 0.95 Penality l1 Train Score 0.9091836734693878 Test Score 0.896507936507
Alpha 0.00012 l1_ratio 0.95 Penality l2 Train Score 0.9676190476190476 Test Score 0.911428571428
Alpha 0.00018 l1_ratio 0 Penality l1 Train Score 0.8865306122448979 Test Score 0.871587301587301
Alpha 0.00018 l1_ratio 0 Penality l2 Train Score 0.9310884353741496 Test Score 0.884603174603174
Alpha 0.00018 l1_ratio 0.03 Penality l1 Train Score 0.8848979591836734 Test Score 0.867777777777
Alpha 0.00018 l1_ratio 0.03 Penality l2 Train Score 0.937891156462585 Test Score 0.891111111111
Alpha 0.00018 l1_ratio 0.05 Penality l1 Train Score 0.8859183673469387 Test Score 0.869523809523
Alpha 0.00018 l1_ratio 0.05 Penality l2 Train Score 0.9338095238095238 Test Score 0.887619047619
Alpha 0.00018 l1_ratio 0.08 Penality l1 Train Score 0.8852380952380953 Test Score 0.869047619047
Alpha 0.00018 l1_ratio 0.08 Penality l2 Train Score 0.9357823129251701 Test Score 0.888888888888
Alpha 0.00018 l1_ratio 0.1 Penality l1 Train Score 0.8846938775510204 Test Score 0.868412698412
Alpha 0.00018 l1_ratio 0.1 Penality l2 Train Score 0.9347619047619048 Test Score 0.888095238095
Alpha 0.00018 l1_ratio 0.15 Penality l1 Train Score 0.8858503401360545 Test Score 0.870793650793
Alpha 0.00018 l1_ratio 0.15 Penality l2 Train Score 0.9362585034013605 Test Score 0.891111111111
Alpha 0.00018 l1_ratio 0.25 Penality l1 Train Score 0.8900680272108844 Test Score 0.874126984126
Alpha 0.00018 l1_ratio 0.25 Penality l2 Train Score 0.932312925170068 Test Score 0.885714285714
Alpha 0.00018 l1_ratio 0.35 Penality l1 Train Score 0.8859863945578231 Test Score 0.870317460317
Alpha 0.00018 l1_ratio 0.35 Penality l2 Train Score 0.9297278911564626 Test Score 0.883809523809
Alpha 0.00018 l1_ratio 0.45 Penality l1 Train Score 0.8850340136054422 Test Score 0.867936507936
Alpha 0.00018 l1_ratio 0.45 Penality l2 Train Score 0.9336734693877551 Test Score 0.888095238095
Alpha 0.00018 l1_ratio 0.55 Penality l1 Train Score 0.886734693877551 Test Score 0.871904761904
Alpha 0.00018 l1_ratio 0.55 Penality l2 Train Score 0.936734693877551 Test Score 0.890476190476
Alpha 0.00018 l1_ratio 0.65 Penality l1 Train Score 0.8866666666666667 Test Score 0.871111111111
Alpha 0.00018 l1_ratio 0.65 Penality l2 Train Score 0.9308843537414966 Test Score 0.884920634920
Alpha 0.00018 l1_ratio 0.75 Penality l1 Train Score 0.8862585034013606 Test Score 0.869841269841
Alpha 0.00018 l1_ratio 0.75 Penality l2 Train Score 0.934625850340136 Test Score 0.888412698412
```

```
Alpha 0.00018 l1_ratio 0.85 Penality l1 Train Score 0.8873469387755102 Test Score 0.870793650
Alpha 0.00018 l1_ratio 0.85 Penality l2 Train Score 0.9358503401360544 Test Score 0.89
Alpha 0.00018 l1_ratio 0.95 Penality l1 Train Score 0.886734693877551 Test Score 0.87158730158
Alpha 0.00018 l1_ratio 0.95 Penality l2 Train Score 0.9342176870748299 Test Score 0.887936507
Alpha 0.00023 l1_ratio 0 Penality l1 Train Score 0.872312925170068 Test Score 0.85190476190476
Alpha 0.00023 l1_ratio 0 Penality l2 Train Score 0.9082993197278911 Test Score 0.869047619047
Alpha 0.00023 l1_ratio 0.03 Penality l1 Train Score 0.8717006802721089 Test Score 0.849682539
Alpha 0.00023 l1_ratio 0.03 Penality l2 Train Score 0.9046258503401361 Test Score 0.866349206
Alpha 0.00023 l1_ratio 0.05 Penality l1 Train Score 0.8714965986394558 Test Score 0.849682539
Alpha 0.00023 l1_ratio 0.05 Penality l2 Train Score 0.9078231292517007 Test Score 0.868412698
Alpha 0.00023 l1_ratio 0.08 Penality l1 Train Score 0.8709523809523809 Test Score 0.850952380
Alpha 0.00023 l1_ratio 0.08 Penality l2 Train Score 0.9056462585034014 Test Score 0.866507936
Alpha 0.00023 l1_ratio 0.1 Penality l1 Train Score 0.867482993197279 Test Score 0.845238095238
Alpha 0.00023 l1_ratio 0.1 Penality l2 Train Score 0.9051020408163265 Test Score 0.866825396825
Alpha 0.00023 l1_ratio 0.15 Penality l1 Train Score 0.8693197278911564 Test Score 0.847142857
Alpha 0.00023 l1_ratio 0.15 Penality l2 Train Score 0.9060544217687074 Test Score 0.867301587
Alpha 0.00023 l1_ratio 0.25 Penality l1 Train Score 0.8701360544217687 Test Score 0.847142857
Alpha 0.00023 l1_ratio 0.25 Penality l2 Train Score 0.9053061224489796 Test Score 0.866825396
Alpha 0.00023 l1_ratio 0.35 Penality l1 Train Score 0.8735374149659864 Test Score 0.852698412
Alpha 0.00023 l1_ratio 0.35 Penality l2 Train Score 0.906734693877551 Test Score 0.868095238095
Alpha 0.00023 l1_ratio 0.45 Penality l1 Train Score 0.8731972789115646 Test Score 0.852222222
Alpha 0.00023 l1_ratio 0.45 Penality l2 Train Score 0.9056462585034014 Test Score 0.866666666
Alpha 0.00023 l1_ratio 0.55 Penality l1 Train Score 0.8721088435374149 Test Score 0.851428571
Alpha 0.00023 l1_ratio 0.55 Penality l2 Train Score 0.9038095238095238 Test Score 0.865238095
Alpha 0.00023 l1_ratio 0.65 Penality l1 Train Score 0.8682312925170068 Test Score 0.847619047
Alpha 0.00023 l1_ratio 0.65 Penality l2 Train Score 0.904421768707483 Test Score 0.866190476190
Alpha 0.00023 l1_ratio 0.75 Penality l1 Train Score 0.8731292517006802 Test Score 0.853333333
Alpha 0.00023 l1_ratio 0.75 Penality l2 Train Score 0.9059183673469388 Test Score 0.866507936
Alpha 0.00023 l1_ratio 0.85 Penality l1 Train Score 0.871156462585034 Test Score 0.849206349206
Alpha 0.00023 l1_ratio 0.85 Penality l2 Train Score 0.9070068027210885 Test Score 0.867301587
Alpha 0.00023 l1_ratio 0.95 Penality l1 Train Score 0.8695918367346939 Test Score 0.849047619
Alpha 0.00023 l1_ratio 0.95 Penality l2 Train Score 0.9082312925170068 Test Score 0.869523809
```

In Random search i didnt got some results wit high l1 ratio and l2 peanality so i dicided to try some low l1 ratios wit same learning rate range and l1 penality. i sisnt get this case in random search. so did some initial investigation above and got some good scores without varince problem also.

```
In [69]: param_grid={'sgdclassifier__penalty':['l1','l2'],
         'sgdclassifier__alpha':[0.00003,0.00005,0.00007,0.00008,0.0001,
                                 0.00012,0.00018,0.00023],
         'sgdclassifier__l1_ratio':[0,0.03,0.05,0.08,0.1,0.15,0.25,0.35,0.45,
                                 0.55,0.65,0.75,0.85,0.95]
                            }
         model_grid_tfidf = GridSearchCV(make_pipeline(TfidfVectorizer(ngram_range=(1,2)),
                                                       SGDClassifier(n_jobs=-1)),
                         param_grid=param_grid,cv=TimeSeriesSplit(n_splits=10),n_jobs=-1)
         model_grid_tfidf.fit(train_df.final_text,train_df.Score)
```

```
In [70]: dict_scores = []
         idx = 0
         for i in model_grid_tfidf.grid_scores_:
             dict_score = []
             dict_score.append(i[0]['sgdclassifier__alpha'])
             dict_score.append(i[0]['sgdclassifier__l1_ratio'])
             dict_score.append(i[0]['sgdclassifier__penalty'])
             dict_score.append(i[1])
             dict_score.append(i[2].std())
             dict_score.append(model_grid_tfidf.cv_results_['mean_train_score'][idx])
             dict_scores.append(dict_score)
             idx = idx + 1
         scores_df = pd.DataFrame(dict_scores,columns=['alpha','l1_ratio','penality','Test_sco
                                                       'Test_std','Train_score'])

In [72]: scores_df.sort_values('Test_score',ascending=False).head(10)
```

Out[72]:

	alpha	l1_ratio	penality	Test_score	Test_std	Train_score
10	0.00003	0.15	l1	0.929282	0.008460	0.976729
0	0.00003	0.00	l1	0.929178	0.010054	0.976633
2	0.00003	0.03	l1	0.929125	0.010393	0.976605
4	0.00003	0.05	l1	0.929073	0.008819	0.976753
12	0.00003	0.25	l1	0.928968	0.008937	0.976537
8	0.00003	0.10	l1	0.928968	0.007552	0.976478
6	0.00003	0.08	l1	0.928916	0.008576	0.976547
16	0.00003	0.45	l1	0.928759	0.008902	0.976561
18	0.00003	0.55	l1	0.928654	0.008387	0.976760
24	0.00003	0.85	l1	0.928444	0.007099	0.977262

Got best scores at alpha = 0.00003, l1_ratio = 0.15, penalty = l1 and mean cv score is 0.929282

```
In [88]: #test scores
         #TFIDF with (1,2) gram with cleaned data
         #tfidf vec
         tf_idf_vect = TfidfVectorizer(ngram_range=(1,2))
         final_counts_train = tf_idf_vect.fit_transform(
                 train_df['final_text'].values)
         #test
         X_test = tf_idf_vect.transform(test_df['final_text'].values)

         model = SGDClassifier(penalty='l1',alpha=0.00003,l1_ratio=0.15)
         model.fit(final_counts_train,train_df.Score)
         #Predicting training data
         train_list = model.predict(final_counts_train)
         #Accuracy score
         score_train = accuracy_score(train_df.Score,train_list)
         #predict test cv
         test_list = model.predict(X_test)
         #Accuracy score
```

```
        score_test = accuracy_score(test_df.Score,test_list)
        #precision
        #precision
        test_precision = precision_score(test_df.Score,test_list)
        #recall
        test_recall = recall_score(test_df.Score,test_list)
        #confusion matrix
        confusion_matrix_test = confusion_matrix(test_df.Score,test_list)
        print("penalty='l1',alpha=0.00003,l1_ratio=0.15")
        print('Train Score', score_train)
        print('Test Score',score_test)
        print('Test Precision',test_precision)
        print('Test Recall',test_recall)
        print('Test ConfusionMatrix',confusion_matrix_test)
```

```
penalty='l1',alpha=0.00003,l1_ratio=0.15
Train Score 0.9569047619047619
Test Score 0.9355555555555556
Test Precision 0.9517659462308908
Test Recall 0.9712210866057019
Test ConfusionMatrix [[1198  366]
 [ 214 7222]]
```

0.0.4 Word2Vec

```
In [19]: #importing
         from gensim.models import Word2Vec
         from gensim.models import KeyedVectors
         import pickle
         import gensim
```

```
In [20]: import gensim
         list_of_sent=[]
         for sent in final_review.final_text.values:
             list_of_sent.append(sent.split())
```

```
In [24]: #word2vec model with 50 dim vector
         w2v_model_50=gensim.models.Word2Vec(list_of_sent,min_count=5,size=50, workers=8)
         #word2vec model with 100 dim vector
         w2v_model_100=gensim.models.Word2Vec(list_of_sent,min_count=5,size=100, workers=8)
         #word2vec model with 300 dim vector
         w2v_model_300=gensim.models.Word2Vec(list_of_sent,min_count=5,size=300, workers=8)
```

```
In [27]: #saving to disk
         pickle.dump(w2v_model_50,open('w2v_model_svm_50.p','wb'))
         pickle.dump(w2v_model_100,open('w2v_model_svm_100.p','wb'))
         pickle.dump(w2v_model_300,open('w2v_model_svm_300.p','wb'))
```

In [21]:
```
#loading from disk
w2v_model_100 = pickle.load(open('w2v_model_svm_100.p','rb'))
w2v_model_50 = pickle.load(open('w2v_model_svm_50.p','rb'))
w2v_model_300 = pickle.load(open('w2v_model_svm_300.p','rb'))
```

Avg Word2Vec

In [16]:
```
# the avg-w2v for each sentence/review is stored in this list
def avg_w2v(list_of_sent,model,d):
    '''
    Returns average of word vectors for
    each sentance with dimension of model given
    '''
    sent_vectors = []
    for sent in list_of_sent: # for each review/sentence
        doc = [word for word in sent if word in model.wv.vocab]
        if doc:
            sent_vec = np.mean(model.wv[doc],axis=0)
        else:
            sent_vec = np.zeros(d)
        sent_vectors.append(sent_vec)
    return sent_vectors
```

In [17]:
```
list_of_sent_train=[]
for sent in train_df.final_text.values:
    list_of_sent_train.append(sent.split())
```

In [18]:
```
#avg word2vec for
sent_vector_avgw2v_300 = avg_w2v(list_of_sent_train,w2v_model_300,300)
#stacking columns
train_avgw2v_300 = np.hstack((sent_vector_avgw2v_300,
            train_df[['HelpfulnessNumerator','HelpfulnessDenominator','Score']]))
column = list(range(0,300))
column.extend(['HelpfulnessNumerator','HelpfulnessDenominator','Score'])
train_df_avgw2v_300 = pd.DataFrame(train_avgw2v_300,columns=column)
```

In [19]:
```
#CountVectorizer for BoW
X_train = train_df_avgw2v_300.iloc[:round(train_df.shape[0]*0.70),:]
X_test_cv = train_df_avgw2v_300.iloc[round(train_df.shape[0]*0.70):,:]
scale = StandardScaler()
X_train_sc = scale.fit_transform(X_train.drop('Score',axis=1))
X_test_cv_sc = scale.transform(X_test_cv.drop('Score',axis=1))
```

In [19]:
```
for i in ParameterGrid({'C':[0.001,0.01,0.1,1,5],
                        'gamma':[0.001,0.008,0.01,0.1,0.5,1,10]}):
    model = SVC(C=i['C'],kernel='rbf',gamma=i['gamma'])
    model.fit(X_train_sc,X_train.Score)
    train_score = model.score(X_train_sc,X_train.Score)
    test_score = model.score(X_test_cv_sc,X_test_cv.Score)
```

```
            print('C',i['C'],'Gamma',i['gamma'],'Train Score',train_score,
                                'Test Score',test_score)
```

```
C 0.001 Gamma 0.001 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 0.008 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 0.01 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 0.1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 0.5 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 10 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 0.001 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 0.008 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 0.01 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 0.1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 0.5 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 10 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 0.001 Train Score 0.9078231292517007 Test Score 0.8917460317460317
C 0.1 Gamma 0.008 Train Score 0.8776190476190476 Test Score 0.8468253968253968
C 0.1 Gamma 0.01 Train Score 0.8648299319727891 Test Score 0.8352380952380952
C 0.1 Gamma 0.1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 0.5 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 10 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 1 Gamma 0.001 Train Score 0.9448299319727891 Test Score 0.9274603174603174
C 1 Gamma 0.008 Train Score 0.98578231292517 Test Score 0.9068253968253969
C 1 Gamma 0.01 Train Score 0.9902721088435374 Test Score 0.8960317460317461
C 1 Gamma 0.1 Train Score 1.0 Test Score 0.8253968253968254
C 1 Gamma 0.5 Train Score 1.0 Test Score 0.8253968253968254
C 1 Gamma 1 Train Score 1.0 Test Score 0.8253968253968254
C 1 Gamma 10 Train Score 1.0 Test Score 0.8253968253968254
C 5 Gamma 0.001 Train Score 0.9637414965986395 Test Score 0.936031746031746
C 5 Gamma 0.008 Train Score 1.0 Test Score 0.9119047619047619
C 5 Gamma 0.01 Train Score 1.0 Test Score 0.9014285714285715
C 5 Gamma 0.1 Train Score 1.0 Test Score 0.8253968253968254
C 5 Gamma 0.5 Train Score 1.0 Test Score 0.8253968253968254
C 5 Gamma 1 Train Score 1.0 Test Score 0.8253968253968254
C 5 Gamma 10 Train Score 1.0 Test Score 0.8253968253968254
```

We can observe that for low values of c we ave ig bias in model and for high values of c we are overfitting to the train data. and for some values of gamma(0.001) and C(1-10) the cv score are better than the others. we do have some generalization error if we are regularizing max also so my be with moderate High C wit low gamma be better for this data.

```
In [20]: train_df_avgw2v_300.to_csv('train_df_avgw2v_300_svm.csv',index=False)
```

```
In [37]: c = [0.8,0.9,1,1.5,3,5,7,10]
         gamma = [0.0005,0.0008,0.00095,0.001,0.003,0.005,0.008]
```

```
model_grid_avgw2v = GridSearchCV(make_pipeline(StandardScaler(),
                                               SVC()),
                                 param_grid={'svc__C': c,'svc__gamma':gamma},
                                cv=TimeSeriesSplit(n_splits=10),n_jobs=-1)
model_grid_avgw2v.fit(train_df_avgw2v_300.drop('Score',axis=1),train_df_avgw2v_300.Sc
```

In [39]:
```
dict_scores = []
idx = 0
for i in model_grid_avgw2v.grid_scores_:
    dict_score = []
    dict_score.append(i[0]['svc__gamma'])
    dict_score.append(i[0]['svc__C'])
    dict_score.append(i[1])
    dict_score.append(i[2].std())
    dict_score.append(model_grid_avgw2v.cv_results_['mean_train_score'][idx])
    dict_scores.append(dict_score)
    idx = idx + 1
scores_df = pd.DataFrame(dict_scores,columns=['gamma','C','Test_score',
                                              'Test_std','Train_score'])
```

In [41]:
```
scores_df.sort_values('Test_score',ascending=False).head(10)
```

Out[41]:

	gamma	C	Test_score	Test_std	Train_score
52	0.00100	10.0	0.933630	0.006818	0.978743
51	0.00095	10.0	0.933578	0.006756	0.977413
50	0.00080	10.0	0.932897	0.006800	0.973149
45	0.00100	7.0	0.932216	0.006664	0.973193
43	0.00080	7.0	0.932006	0.007466	0.968160
44	0.00095	7.0	0.931797	0.007015	0.971904
38	0.00100	5.0	0.931797	0.007716	0.968227
37	0.00095	5.0	0.931744	0.007464	0.967187
49	0.00050	10.0	0.931640	0.006226	0.962913
39	0.00300	5.0	0.931535	0.004997	0.993881

In [50]:
```
c = [10,20,30,40,50,60,70,80,90,100]
gamma = [0.00095,0.001]
model_grid_avgw2v2 = GridSearchCV(make_pipeline(StandardScaler(),
                                               SVC()),
                                  param_grid={'svc__C': c,'svc__gamma':gamma},
                                 cv=TimeSeriesSplit(n_splits=10),n_jobs=-1)
model_grid_avgw2v2.fit(train_df_avgw2v_300.drop('Score',axis=1),train_df_avgw2v_300.S
```

In [51]:
```
dict_scores = []
idx = 0
for i in model_grid_avgw2v2.grid_scores_:
    dict_score = []
    dict_score.append(i[0]['svc__gamma'])
    dict_score.append(i[0]['svc__C'])
    dict_score.append(i[1])
```

```
        dict_score.append(i[2].std())
        dict_score.append(model_grid_avgw2v2.cv_results_['mean_train_score'][idx])
        dict_scores.append(dict_score)
        idx = idx + 1
    scores_df2 = pd.DataFrame(dict_scores,columns=['gamma','C','Test_score',
                                                   'Test_std','Train_score'])
```

In [53]:
```
scores_df2.sort_values('Test_score',ascending=False).head(10)
```

Out[53]:

	gamma	C	Test_score	Test_std	Train_score
1	0.00100	10	0.933630	0.006818	0.978743
0	0.00095	10	0.933578	0.006756	0.977413
3	0.00100	20	0.932111	0.006485	0.987762
2	0.00095	20	0.931640	0.007013	0.986663
5	0.00100	30	0.931221	0.006858	0.992037
4	0.00095	30	0.931116	0.006455	0.991255
6	0.00095	40	0.930906	0.006735	0.993642
7	0.00100	40	0.930592	0.007110	0.994460
8	0.00095	50	0.930016	0.007068	0.995593
9	0.00100	50	0.929387	0.006629	0.996349

In [76]:
```
from scipy.stats import uniform
model_random_avgw2v = RandomizedSearchCV(make_pipeline(StandardScaler(),SVC()),
                      param_distributions={'svc__C': uniform(loc=0,scale=13),
                      'svc__gamma':uniform(loc=0.0008,scale=0.004)},n_iter=25,
                            cv=TimeSeriesSplit(n_splits=10),n_jobs=-1)
model_random_avgw2v.fit(train_df_avgw2v_300.drop('Score',axis=1),train_df_avgw2v_300.
```

In [77]:
```
dict_scores = []
idx = 0
for i in model_random_avgw2v.grid_scores_:
    dict_score = []
    dict_score.append(i[0]['svc__gamma'])
    dict_score.append(i[0]['svc__C'])
    dict_score.append(i[1])
    dict_score.append(i[2].std())
    dict_score.append(model_random_avgw2v.cv_results_['mean_train_score'][idx])
    dict_scores.append(dict_score)
    idx = idx + 1
scores_df1 = pd.DataFrame(dict_scores,columns=['gamma','C','Test_score',
                                               'Test_std','Train_score'])
```

In [78]:
```
scores_df1.sort_values('Test_score',ascending=False).head(10)
```

Out[78]:

	gamma	C	Test_score	Test_std	Train_score
0	0.001337	7.135116	0.932792	0.006643	0.980713
21	0.001729	6.847431	0.932740	0.006190	0.986392
5	0.001961	4.655458	0.932635	0.006548	0.983589
19	0.001799	6.445765	0.932583	0.006304	0.986443

```
22  0.001156  12.931930    0.932478  0.006379     0.985614
13  0.001583  10.511907    0.932373  0.006640     0.990104
1   0.002504   3.154084    0.932216  0.006163     0.983583
12  0.002473   5.143757    0.932163  0.006208     0.990524
18  0.001417  12.460012    0.932111  0.006425     0.989660
16  0.002093   2.559956    0.931954  0.006908     0.975369
```

Best cv score for 10 fold cv got at `gamma = 0.00100, C = 10` and mean cv score is `0.933630`

In [82]:
```
#testscore
list_of_sent_train=[]
for sent in train_df.final_text.values:
    list_of_sent_train.append(sent.split())
#avg word2vec for
sent_vector_avgw2v_300 = avg_w2v(list_of_sent_train,w2v_model_300,300)
#stacking columns
train_avgw2v_300 = np.hstack((sent_vector_avgw2v_300,
            train_df[['HelpfulnessNumerator','HelpfulnessDenominator','Score']]))
column = list(range(0,300))
column.extend(['HelpfulnessNumerator','HelpfulnessDenominator','Score'])
train_df_avgw2v_300 = pd.DataFrame(train_avgw2v_300,columns=column)


list_of_sent_test=[]
for sent in test_df.final_text.values:
    list_of_sent_test.append(sent.split())
#avg word2vec for
sent_vector_avgw2v_300_test = avg_w2v(list_of_sent_test,w2v_model_300,300)
#stacking columns
test_avgw2v_300 = np.hstack((sent_vector_avgw2v_300_test,
            test_df[['HelpfulnessNumerator','HelpfulnessDenominator','Score']]))
column = list(range(0,300))
column.extend(['HelpfulnessNumerator','HelpfulnessDenominator','Score'])
test_df_avgw2v_300 = pd.DataFrame(test_avgw2v_300,columns=column)


scale = StandardScaler()
X_train_sc = scale.fit_transform(train_df_avgw2v_300.drop('Score',axis=1))
X_test_cv_sc = scale.transform(test_df_avgw2v_300.drop('Score',axis=1))


model = SVC(C=10,kernel='rbf',gamma=0.00100)
model.fit(X_train_sc,train_df.Score)
#Predicting training data
train_list = model.predict(X_train_sc)
#Accuracy score
score_train = accuracy_score(train_df.Score,train_list)
#predict test cv
```

```
        test_list = model.predict(X_test_cv_sc)
        #Accuracy score
        score_test = accuracy_score(test_df.Score,test_list)
        #precision
        #precision
        test_precision = precision_score(test_df.Score,test_list)
        #recall
        test_recall = recall_score(test_df.Score,test_list)
        #confusion matrix
        confusion_matrix_test = confusion_matrix(test_df.Score,test_list)
        print('C' ,10,'gamma',0.00100)
        print('Train Score', score_train)
        print('Test Score',score_test)
        print('Test Precision',test_precision)
        print('Test Recall',test_recall)
        print('Test ConfusionMatrix',confusion_matrix_test)
        print('No of support vectors for each class',model.n_support_)
```

```
C 10 gamma 0.001
Train Score 0.9725714285714285
Test Score 0.9332222222222222
Test Precision 0.9459741615555266
Test Recall 0.9748520710059172
Test ConfusionMatrix [[1150  414]
 [ 187 7249]]
No of support vectors for each class [1693 2078]
```

SGD Classifier

```
In [27]: for i in ParameterGrid({'alpha':[0.00005,0.00008,0.0001,0.00012],
                                'l1_ratio':[0,0.03,0.05,0.08,0.1,0.15,0.25,0.35,
                                            0.45,0.55,0.65,0.75,0.85,0.95],
                                'penality':['l1','l2','elasticnet']}):
            model = SGDClassifier(penalty=i['penality'],alpha=i['alpha'],l1_ratio=i['l1_ratio
            model.fit(X_train_sc,X_train.Score)
            train_score = model.score(X_train_sc,X_train.Score)
            test_score = model.score(X_test_cv_sc,X_test_cv.Score)
            print('Alpha',i['alpha'],'l1_ratio',i['l1_ratio'],'Penality',i['penality'],
                  'Train Score',train_score,'Test Score',test_score)
```

```
Alpha 5e-05 l1_ratio 0 Penality l1 Train Score 0.9415646258503402 Test Score 0.929365079365079
Alpha 5e-05 l1_ratio 0 Penality l2 Train Score 0.9393197278911565 Test Score 0.927936507936508
Alpha 5e-05 l1_ratio 0 Penality elasticnet Train Score 0.9393197278911565 Test Score 0.92793650
Alpha 5e-05 l1_ratio 0.03 Penality l1 Train Score 0.9415646258503402 Test Score 0.92936507936
Alpha 5e-05 l1_ratio 0.03 Penality l2 Train Score 0.9393197278911565 Test Score 0.92793650793
Alpha 5e-05 l1_ratio 0.03 Penality elasticnet Train Score 0.9377551020408164 Test Score 0.9250
Alpha 5e-05 l1_ratio 0.05 Penality l1 Train Score 0.9415646258503402 Test Score 0.92936507936
```

```
Alpha 5e-05 l1_ratio 0.05 Penality l2 Train Score 0.9393197278911565 Test Score 0.927936507936
Alpha 5e-05 l1_ratio 0.05 Penality elasticnet Train Score 0.9386394557823129 Test Score 0.9258
Alpha 5e-05 l1_ratio 0.08 Penality l1 Train Score 0.9415646258503402 Test Score 0.92936507936
Alpha 5e-05 l1_ratio 0.08 Penality l2 Train Score 0.9393197278911565 Test Score 0.927936507936
Alpha 5e-05 l1_ratio 0.08 Penality elasticnet Train Score 0.9385034013605442 Test Score 0.9255
Alpha 5e-05 l1_ratio 0.1 Penality l1 Train Score 0.9415646258503402 Test Score 0.92936507936
Alpha 5e-05 l1_ratio 0.1 Penality l2 Train Score 0.9393197278911565 Test Score 0.92793650793
Alpha 5e-05 l1_ratio 0.1 Penality elasticnet Train Score 0.9387074829931973 Test Score 0.9277
Alpha 5e-05 l1_ratio 0.15 Penality l1 Train Score 0.9415646258503402 Test Score 0.92936507936
Alpha 5e-05 l1_ratio 0.15 Penality l2 Train Score 0.9393197278911565 Test Score 0.927936507936
Alpha 5e-05 l1_ratio 0.15 Penality elasticnet Train Score 0.9374829931972789 Test Score 0.9230
Alpha 5e-05 l1_ratio 0.25 Penality l1 Train Score 0.9415646258503402 Test Score 0.92936507936
Alpha 5e-05 l1_ratio 0.25 Penality l2 Train Score 0.9393197278911565 Test Score 0.927936507936
Alpha 5e-05 l1_ratio 0.25 Penality elasticnet Train Score 0.939047619047619 Test Score 0.92619
Alpha 5e-05 l1_ratio 0.35 Penality l1 Train Score 0.9415646258503402 Test Score 0.92936507936
Alpha 5e-05 l1_ratio 0.35 Penality l2 Train Score 0.9393197278911565 Test Score 0.927936507936
Alpha 5e-05 l1_ratio 0.35 Penality elasticnet Train Score 0.9387074829931973 Test Score 0.9265
Alpha 5e-05 l1_ratio 0.45 Penality l1 Train Score 0.9415646258503402 Test Score 0.92936507936
Alpha 5e-05 l1_ratio 0.45 Penality l2 Train Score 0.9393197278911565 Test Score 0.927936507936
Alpha 5e-05 l1_ratio 0.45 Penality elasticnet Train Score 0.9392517006802721 Test Score 0.9273
Alpha 5e-05 l1_ratio 0.55 Penality l1 Train Score 0.9415646258503402 Test Score 0.92936507936
Alpha 5e-05 l1_ratio 0.55 Penality l2 Train Score 0.9393197278911565 Test Score 0.927936507936
Alpha 5e-05 l1_ratio 0.55 Penality elasticnet Train Score 0.941156462585034 Test Score 0.9298
Alpha 5e-05 l1_ratio 0.65 Penality l1 Train Score 0.9415646258503402 Test Score 0.92936507936
Alpha 5e-05 l1_ratio 0.65 Penality l2 Train Score 0.9393197278911565 Test Score 0.927936507936
Alpha 5e-05 l1_ratio 0.65 Penality elasticnet Train Score 0.9406802721088435 Test Score 0.9263
Alpha 5e-05 l1_ratio 0.75 Penality l1 Train Score 0.9415646258503402 Test Score 0.92936507936
Alpha 5e-05 l1_ratio 0.75 Penality l2 Train Score 0.9393197278911565 Test Score 0.927936507936
Alpha 5e-05 l1_ratio 0.75 Penality elasticnet Train Score 0.9422448979591836 Test Score 0.9292
Alpha 5e-05 l1_ratio 0.85 Penality l1 Train Score 0.9415646258503402 Test Score 0.92936507936
Alpha 5e-05 l1_ratio 0.85 Penality l2 Train Score 0.9393197278911565 Test Score 0.927936507936
Alpha 5e-05 l1_ratio 0.85 Penality elasticnet Train Score 0.9417006802721088 Test Score 0.9290
Alpha 5e-05 l1_ratio 0.95 Penality l1 Train Score 0.9415646258503402 Test Score 0.92936507936
Alpha 5e-05 l1_ratio 0.95 Penality l2 Train Score 0.9393197278911565 Test Score 0.927936507936
Alpha 5e-05 l1_ratio 0.95 Penality elasticnet Train Score 0.943469387755102 Test Score 0.9274
Alpha 8e-05 l1_ratio 0 Penality l1 Train Score 0.9438095238095238 Test Score 0.928571428571428
Alpha 8e-05 l1_ratio 0 Penality l2 Train Score 0.9384353741496598 Test Score 0.924603174603174
Alpha 8e-05 l1_ratio 0 Penality elasticnet Train Score 0.9384353741496598 Test Score 0.9246031
Alpha 8e-05 l1_ratio 0.03 Penality l1 Train Score 0.9438095238095238 Test Score 0.92857142857
Alpha 8e-05 l1_ratio 0.03 Penality l2 Train Score 0.9384353741496598 Test Score 0.92460317460
Alpha 8e-05 l1_ratio 0.03 Penality elasticnet Train Score 0.937687074829932 Test Score 0.92365
Alpha 8e-05 l1_ratio 0.05 Penality l1 Train Score 0.9438095238095238 Test Score 0.92857142857
Alpha 8e-05 l1_ratio 0.05 Penality l2 Train Score 0.9384353741496598 Test Score 0.92460317460
Alpha 8e-05 l1_ratio 0.05 Penality elasticnet Train Score 0.9385034013605442 Test Score 0.9246
Alpha 8e-05 l1_ratio 0.08 Penality l1 Train Score 0.9438095238095238 Test Score 0.92857142857
Alpha 8e-05 l1_ratio 0.08 Penality l2 Train Score 0.9384353741496598 Test Score 0.92460317460
Alpha 8e-05 l1_ratio 0.08 Penality elasticnet Train Score 0.9373469387755102 Test Score 0.9233
Alpha 8e-05 l1_ratio 0.1 Penality l1 Train Score 0.9438095238095238 Test Score 0.928571428571
```

```
Alpha 8e-05 l1_ratio 0.1 Penality l2 Train Score 0.9384353741496598 Test Score 0.9246031746031
Alpha 8e-05 l1_ratio 0.1 Penality elasticnet Train Score 0.9389115646258503 Test Score 0.923968
Alpha 8e-05 l1_ratio 0.15 Penality l1 Train Score 0.9438095238095238 Test Score 0.928571428571
Alpha 8e-05 l1_ratio 0.15 Penality l2 Train Score 0.9384353741496598 Test Score 0.924603174603
Alpha 8e-05 l1_ratio 0.15 Penality elasticnet Train Score 0.9386394557823129 Test Score 0.9239
Alpha 8e-05 l1_ratio 0.25 Penality l1 Train Score 0.9438095238095238 Test Score 0.928571428571
Alpha 8e-05 l1_ratio 0.25 Penality l2 Train Score 0.9384353741496598 Test Score 0.924603174603
Alpha 8e-05 l1_ratio 0.25 Penality elasticnet Train Score 0.939047619047619 Test Score 0.92571
Alpha 8e-05 l1_ratio 0.35 Penality l1 Train Score 0.9438095238095238 Test Score 0.928571428571
Alpha 8e-05 l1_ratio 0.35 Penality l2 Train Score 0.9384353741496598 Test Score 0.924603174603
Alpha 8e-05 l1_ratio 0.35 Penality elasticnet Train Score 0.9385714285714286 Test Score 0.9255
Alpha 8e-05 l1_ratio 0.45 Penality l1 Train Score 0.9438095238095238 Test Score 0.928571428571
Alpha 8e-05 l1_ratio 0.45 Penality l2 Train Score 0.9384353741496598 Test Score 0.924603174603
Alpha 8e-05 l1_ratio 0.45 Penality elasticnet Train Score 0.9403401360544218 Test Score 0.9265
Alpha 8e-05 l1_ratio 0.55 Penality l1 Train Score 0.9438095238095238 Test Score 0.928571428571
Alpha 8e-05 l1_ratio 0.55 Penality l2 Train Score 0.9384353741496598 Test Score 0.924603174603
Alpha 8e-05 l1_ratio 0.55 Penality elasticnet Train Score 0.9396598639455782 Test Score 0.9269
Alpha 8e-05 l1_ratio 0.65 Penality l1 Train Score 0.9438095238095238 Test Score 0.928571428571
Alpha 8e-05 l1_ratio 0.65 Penality l2 Train Score 0.9384353741496598 Test Score 0.924603174603
Alpha 8e-05 l1_ratio 0.65 Penality elasticnet Train Score 0.9417006802721088 Test Score 0.9280
Alpha 8e-05 l1_ratio 0.75 Penality l1 Train Score 0.9438095238095238 Test Score 0.928571428571
Alpha 8e-05 l1_ratio 0.75 Penality l2 Train Score 0.9384353741496598 Test Score 0.924603174603
Alpha 8e-05 l1_ratio 0.75 Penality elasticnet Train Score 0.9415646258503402 Test Score 0.9290
Alpha 8e-05 l1_ratio 0.85 Penality l1 Train Score 0.9438095238095238 Test Score 0.928571428571
Alpha 8e-05 l1_ratio 0.85 Penality l2 Train Score 0.9384353741496598 Test Score 0.924603174603
Alpha 8e-05 l1_ratio 0.85 Penality elasticnet Train Score 0.9398639455782313 Test Score 0.9280
Alpha 8e-05 l1_ratio 0.95 Penality l1 Train Score 0.9438095238095238 Test Score 0.928571428571
Alpha 8e-05 l1_ratio 0.95 Penality l2 Train Score 0.9384353741496598 Test Score 0.924603174603
Alpha 8e-05 l1_ratio 0.95 Penality elasticnet Train Score 0.9425850340136055 Test Score 0.9273
Alpha 0.0001 l1_ratio 0 Penality l1 Train Score 0.9437414965986395 Test Score 0.927936507936508
Alpha 0.0001 l1_ratio 0 Penality l2 Train Score 0.9366666666666666 Test Score 0.922857142857142
Alpha 0.0001 l1_ratio 0 Penality elasticnet Train Score 0.9366666666666666 Test Score 0.92285
Alpha 0.0001 l1_ratio 0.03 Penality l1 Train Score 0.9437414965986395 Test Score 0.92793650793
Alpha 0.0001 l1_ratio 0.03 Penality l2 Train Score 0.9366666666666666 Test Score 0.92285714285
Alpha 0.0001 l1_ratio 0.03 Penality elasticnet Train Score 0.9374829931972789 Test Score 0.922
Alpha 0.0001 l1_ratio 0.05 Penality l1 Train Score 0.9437414965986395 Test Score 0.92793650793
Alpha 0.0001 l1_ratio 0.05 Penality l2 Train Score 0.9366666666666666 Test Score 0.92285714285
Alpha 0.0001 l1_ratio 0.05 Penality elasticnet Train Score 0.9378231292517006 Test Score 0.924
Alpha 0.0001 l1_ratio 0.08 Penality l1 Train Score 0.9437414965986395 Test Score 0.92793650793
Alpha 0.0001 l1_ratio 0.08 Penality l2 Train Score 0.9366666666666666 Test Score 0.92285714285
Alpha 0.0001 l1_ratio 0.08 Penality elasticnet Train Score 0.9373469387755102 Test Score 0.923
Alpha 0.0001 l1_ratio 0.1 Penality l1 Train Score 0.9437414965986395 Test Score 0.927936507936
Alpha 0.0001 l1_ratio 0.1 Penality l2 Train Score 0.9366666666666666 Test Score 0.922857142857
Alpha 0.0001 l1_ratio 0.1 Penality elasticnet Train Score 0.9380272108843537 Test Score 0.9236
Alpha 0.0001 l1_ratio 0.15 Penality l1 Train Score 0.9437414965986395 Test Score 0.92793650793
Alpha 0.0001 l1_ratio 0.15 Penality l2 Train Score 0.9366666666666666 Test Score 0.92285714285
Alpha 0.0001 l1_ratio 0.15 Penality elasticnet Train Score 0.9380272108843537 Test Score 0.923
Alpha 0.0001 l1_ratio 0.25 Penality l1 Train Score 0.9437414965986395 Test Score 0.92793650793
```

```
Alpha 0.0001 l1_ratio 0.25 Penality l2 Train Score 0.9366666666666666 Test Score 0.92285714285
Alpha 0.0001 l1_ratio 0.25 Penality elasticnet Train Score 0.9382993197278912 Test Score 0.924
Alpha 0.0001 l1_ratio 0.35 Penality l1 Train Score 0.9437414965986395 Test Score 0.92793650793
Alpha 0.0001 l1_ratio 0.35 Penality l2 Train Score 0.9366666666666666 Test Score 0.92285714285
Alpha 0.0001 l1_ratio 0.35 Penality elasticnet Train Score 0.9391836734693878 Test Score 0.923
Alpha 0.0001 l1_ratio 0.45 Penality l1 Train Score 0.9437414965986395 Test Score 0.92793650793
Alpha 0.0001 l1_ratio 0.45 Penality l2 Train Score 0.9366666666666666 Test Score 0.92285714285
Alpha 0.0001 l1_ratio 0.45 Penality elasticnet Train Score 0.940204081632653 Test Score 0.9255
Alpha 0.0001 l1_ratio 0.55 Penality l1 Train Score 0.9437414965986395 Test Score 0.92793650793
Alpha 0.0001 l1_ratio 0.55 Penality l2 Train Score 0.9366666666666666 Test Score 0.92285714285
Alpha 0.0001 l1_ratio 0.55 Penality elasticnet Train Score 0.9401360544217687 Test Score 0.927
Alpha 0.0001 l1_ratio 0.65 Penality l1 Train Score 0.9437414965986395 Test Score 0.92793650793
Alpha 0.0001 l1_ratio 0.65 Penality l2 Train Score 0.9366666666666666 Test Score 0.92285714285
Alpha 0.0001 l1_ratio 0.65 Penality elasticnet Train Score 0.9406802721088435 Test Score 0.926
Alpha 0.0001 l1_ratio 0.75 Penality l1 Train Score 0.9437414965986395 Test Score 0.92793650793
Alpha 0.0001 l1_ratio 0.75 Penality l2 Train Score 0.9366666666666666 Test Score 0.92285714285
Alpha 0.0001 l1_ratio 0.75 Penality elasticnet Train Score 0.9400680272108843 Test Score 0.927
Alpha 0.0001 l1_ratio 0.85 Penality l1 Train Score 0.9437414965986395 Test Score 0.92793650793
Alpha 0.0001 l1_ratio 0.85 Penality l2 Train Score 0.9366666666666666 Test Score 0.92285714285
Alpha 0.0001 l1_ratio 0.85 Penality elasticnet Train Score 0.9415646258503402 Test Score 0.927
Alpha 0.0001 l1_ratio 0.95 Penality l1 Train Score 0.9437414965986395 Test Score 0.92793650793
Alpha 0.0001 l1_ratio 0.95 Penality l2 Train Score 0.9366666666666666 Test Score 0.92285714285
Alpha 0.0001 l1_ratio 0.95 Penality elasticnet Train Score 0.9410884353741497 Test Score 0.926
Alpha 0.00012 l1_ratio 0 Penality l1 Train Score 0.9426530612244898 Test Score 0.92730158730158
Alpha 0.00012 l1_ratio 0 Penality l2 Train Score 0.9385714285714286 Test Score 0.92396825396825
Alpha 0.00012 l1_ratio 0 Penality elasticnet Train Score 0.9385714285714286 Test Score 0.923968
Alpha 0.00012 l1_ratio 0.03 Penality l1 Train Score 0.9426530612244898 Test Score 0.9273015873
Alpha 0.00012 l1_ratio 0.03 Penality l2 Train Score 0.9385714285714286 Test Score 0.923968253
Alpha 0.00012 l1_ratio 0.03 Penality elasticnet Train Score 0.9372789115646258 Test Score 0.92
Alpha 0.00012 l1_ratio 0.05 Penality l1 Train Score 0.9426530612244898 Test Score 0.9273015873
Alpha 0.00012 l1_ratio 0.05 Penality l2 Train Score 0.9385714285714286 Test Score 0.923968253
Alpha 0.00012 l1_ratio 0.05 Penality elasticnet Train Score 0.9374149659863945 Test Score 0.92
Alpha 0.00012 l1_ratio 0.08 Penality l1 Train Score 0.9426530612244898 Test Score 0.9273015873
Alpha 0.00012 l1_ratio 0.08 Penality l2 Train Score 0.9385714285714286 Test Score 0.923968253
Alpha 0.00012 l1_ratio 0.08 Penality elasticnet Train Score 0.9379591836734694 Test Score 0.923
Alpha 0.00012 l1_ratio 0.1 Penality l1 Train Score 0.9426530612244898 Test Score 0.927301587301
Alpha 0.00012 l1_ratio 0.1 Penality l2 Train Score 0.9385714285714286 Test Score 0.923968253968
Alpha 0.00012 l1_ratio 0.1 Penality elasticnet Train Score 0.9379591836734694 Test Score 0.923
Alpha 0.00012 l1_ratio 0.15 Penality l1 Train Score 0.9426530612244898 Test Score 0.9273015873
Alpha 0.00012 l1_ratio 0.15 Penality l2 Train Score 0.9385714285714286 Test Score 0.923968253
Alpha 0.00012 l1_ratio 0.15 Penality elasticnet Train Score 0.9386394557823129 Test Score 0.923
Alpha 0.00012 l1_ratio 0.25 Penality l1 Train Score 0.9426530612244898 Test Score 0.9273015873
Alpha 0.00012 l1_ratio 0.25 Penality l2 Train Score 0.9385714285714286 Test Score 0.923968253
Alpha 0.00012 l1_ratio 0.25 Penality elasticnet Train Score 0.9379591836734694 Test Score 0.923
Alpha 0.00012 l1_ratio 0.35 Penality l1 Train Score 0.9426530612244898 Test Score 0.9273015873
Alpha 0.00012 l1_ratio 0.35 Penality l2 Train Score 0.9385714285714286 Test Score 0.923968253
Alpha 0.00012 l1_ratio 0.35 Penality elasticnet Train Score 0.9382993197278912 Test Score 0.924
Alpha 0.00012 l1_ratio 0.45 Penality l1 Train Score 0.9426530612244898 Test Score 0.9273015873
```

```
Alpha 0.00012 l1_ratio 0.45 Penality l2 Train Score 0.9385714285714286 Test Score 0.9239682539
Alpha 0.00012 l1_ratio 0.45 Penality elasticnet Train Score 0.9382993197278912 Test Score 0.925
Alpha 0.00012 l1_ratio 0.55 Penality l1 Train Score 0.9426530612244898 Test Score 0.9273015873
Alpha 0.00012 l1_ratio 0.55 Penality l2 Train Score 0.9385714285714286 Test Score 0.9239682539
Alpha 0.00012 l1_ratio 0.55 Penality elasticnet Train Score 0.9393197278911565 Test Score 0.926
Alpha 0.00012 l1_ratio 0.65 Penality l1 Train Score 0.9426530612244898 Test Score 0.9273015873
Alpha 0.00012 l1_ratio 0.65 Penality l2 Train Score 0.9385714285714286 Test Score 0.9239682539
Alpha 0.00012 l1_ratio 0.65 Penality elasticnet Train Score 0.9402721088435374 Test Score 0.927
Alpha 0.00012 l1_ratio 0.75 Penality l1 Train Score 0.9426530612244898 Test Score 0.9273015873
Alpha 0.00012 l1_ratio 0.75 Penality l2 Train Score 0.9385714285714286 Test Score 0.9239682539
Alpha 0.00012 l1_ratio 0.75 Penality elasticnet Train Score 0.9406122448979591 Test Score 0.928
Alpha 0.00012 l1_ratio 0.85 Penality l1 Train Score 0.9426530612244898 Test Score 0.9273015873
Alpha 0.00012 l1_ratio 0.85 Penality l2 Train Score 0.9385714285714286 Test Score 0.9239682539
Alpha 0.00012 l1_ratio 0.85 Penality elasticnet Train Score 0.9415646258503402 Test Score 0.926
Alpha 0.00012 l1_ratio 0.95 Penality l1 Train Score 0.9426530612244898 Test Score 0.9273015873
Alpha 0.00012 l1_ratio 0.95 Penality l2 Train Score 0.9385714285714286 Test Score 0.9239682539
Alpha 0.00012 l1_ratio 0.95 Penality elasticnet Train Score 0.9427210884353742 Test Score 0.928
```

In [44]:
```
param_grid={'sgdclassifier__penalty':['l1','l2','elasticnet'],
'sgdclassifier__alpha':[0.00003,0.00005,0.00007,0.00008,0.0001,
                        0.00012,0.00018,0.00023],
'sgdclassifier__l1_ratio':[0,0.03,0.05,0.08,0.1,0.15,0.25,0.35,
                           0.45,0.55,0.65,0.75,0.85,0.95]}
model_grid_avgw2v = GridSearchCV(make_pipeline(StandardScaler(),
                                               SGDClassifier(n_jobs=-1)),
                                    param_grid=param_grid,
                                 cv=TimeSeriesSplit(n_splits=10),
                                                    n_jobs=-1)
model_grid_avgw2v.fit(train_df_avgw2v_300.drop('Score',axis=1),train_df_avgw2v_300.Sc
```

In [48]:
```
dict_scores = []
idx = 0
for i in model_grid_avgw2v.grid_scores_:
    dict_score = []
    dict_score.append(i[0]['sgdclassifier__alpha'])
    dict_score.append(i[0]['sgdclassifier__l1_ratio'])
    dict_score.append(i[0]['sgdclassifier__penalty'])
    dict_score.append(i[1])
    dict_score.append(i[2].std())
    dict_score.append(model_grid_avgw2v.cv_results_['mean_train_score'][idx])
    dict_scores.append(dict_score)
    idx = idx + 1
scores_df = pd.DataFrame(dict_scores,columns=['alpha','l1_ratio','penalty','Test_score
                                              'Test_std','Train_score'])
```

In [49]:
```
scores_df.sort_values('Test_score',ascending=False).head(10)
```

Out[49]:

	alpha	l1_ratio	penalty	Test_score	Test_std	Train_score
228	0.00012	0.25	l1	0.925406	0.005465	0.949701

231	0.00012	0.35	l1	0.925354	0.008048	0.949089
251	0.00012	0.95	elasticnet	0.925039	0.007527	0.948532
213	0.00012	0.03	l1	0.924987	0.006550	0.948186
66	0.00005	0.45	l1	0.924935	0.005871	0.949894
234	0.00012	0.45	l1	0.924830	0.006851	0.948310
186	0.00010	0.25	l1	0.924725	0.006626	0.948830
138	0.00008	0.10	l1	0.924673	0.004765	0.950189
177	0.00010	0.08	l1	0.924515	0.005506	0.948358
79	0.00005	0.85	l2	0.924463	0.005463	0.943389

Got best cv mean score at alpha = 0.00012,l1_ratio= 0.25 penlty = l1 and mean cv score is 0.925406

In [68]:
```
print('With SGD')
#testscore
list_of_sent_train=[]
for sent in train_df.final_text.values:
    list_of_sent_train.append(sent.split())
#avg word2vec for
sent_vector_avgw2v_300 = avg_w2v(list_of_sent_train,w2v_model_300,300)
#stacking columns
train_avgw2v_300 = np.hstack((sent_vector_avgw2v_300,
            train_df[['HelpfulnessNumerator','HelpfulnessDenominator','Score']]))
column = list(range(0,300))
column.extend(['HelpfulnessNumerator','HelpfulnessDenominator','Score'])
train_df_avgw2v_300 = pd.DataFrame(train_avgw2v_300,columns=column)


list_of_sent_test=[]
for sent in test_df.final_text.values:
    list_of_sent_test.append(sent.split())
#avg word2vec for
sent_vector_avgw2v_300_test = avg_w2v(list_of_sent_test,w2v_model_300,300)
#stacking columns
test_avgw2v_300 = np.hstack((sent_vector_avgw2v_300_test,
            test_df[['HelpfulnessNumerator','HelpfulnessDenominator','Score']]))
column = list(range(0,300))
column.extend(['HelpfulnessNumerator','HelpfulnessDenominator','Score'])
test_df_avgw2v_300 = pd.DataFrame(test_avgw2v_300,columns=column)


scale = StandardScaler()
X_train_sc = scale.fit_transform(train_df_avgw2v_300.drop('Score',axis=1))
X_test_cv_sc = scale.transform(test_df_avgw2v_300.drop('Score',axis=1))


model =  SGDClassifier(penalty='l1',alpha=0.00012,l1_ratio=0.25,random_state=25)
model.fit(X_train_sc,train_df.Score)
```

```
        #Predicting training data
        train_list = model.predict(X_train_sc)
        #Accuracy score
        score_train = accuracy_score(train_df.Score,train_list)
        #predict test cv
        test_list = model.predict(X_test_cv_sc)
        #Accuracy score
        score_test = accuracy_score(test_df.Score,test_list)
        #precision
        #precision
        test_precision = precision_score(test_df.Score,test_list)
        #recall
        test_recall = recall_score(test_df.Score,test_list)
        #confusion matrix
        confusion_matrix_test = confusion_matrix(test_df.Score,test_list)
        print("penalty='l1',alpha=0.00012,l1_ratio=0.25")
        print('Train Score', score_train)
        print('Test Score',score_test)
        print('Test Precision',test_precision)
        print('Test Recall',test_recall)
        print('Test ConfusionMatrix',confusion_matrix_test)
```

```
With SGD
penalty='l1',alpha=0.00012,l1_ratio=0.25
Train Score 0.9392380952380952
Test Score 0.9287777777777778
Test Precision 0.9474516001580403
Test Recall 0.9674556213017751
Test ConfusionMatrix [[1165  399]
 [ 242 7194]]
```

0.0.5 Tf-Idf Word2Vec

In [22]:
```
from sklearn.base import BaseEstimator, TransformerMixin

class TfidfWeightedWord2Vec(BaseEstimator, TransformerMixin):
    '''
    Class for Tfidf Weighted Word2Vec Calculations
    '''
    def __init__(self, word2vec):
        self.word2vec = word2vec
        self.word2weight = None
        self.dim = word2vec.vector_size
        self.tfidf = None

    def fit(self, X, y=None):
        tfidf = TfidfVectorizer()
```

```
        tfidf.fit(X[:,0])
        self.tfidf = tfidf
        #print(self.word2vec.wv.vocab.keys())
        return self

    def tf_idf_W2V(self,feature_names,tf_idf_trans_arr,list_of_sent):
        '''
        tfidf weighted word2vec calculation
        '''
        import operator
        dict_tfidf = {k: v for v, k in enumerate(feature_names)}
        sent_vectors = []
        i = 0
        for sent in list_of_sent: # for each review/sentence
            doc = [word for word in sent if word in self.word2vec.wv.vocab.keys()]
            if doc:
                #itemgetter
                f = operator.itemgetter(*doc)
                try:
                    #itemgetter from dict
                    final = f(dict_tfidf)
                    final = tf_idf_trans_arr[i,final]
                    #converting to dense
                    final = final.toarray()
                    #converting to diagnol matrix for multiplication
                    final= np.diag(final[0])
                    sent_vec = np.dot(final,np.array(self.word2vec.wv[doc]))
                    #tfidf weighted word to vec
                    sent_vec = np.sum(sent_vec,axis=0) / np.sum(final)
                except:
                    sent_vec = np.zeros(self.dim)
            else:
                sent_vec = np.zeros(self.dim)
            sent_vectors.append(sent_vec)
            i = i+1
        return sent_vectors

    def transform(self, X):
        #transform data
        tf_idf_trans_arr = self.tfidf.transform(X[:,0])
        feature_names = self.tfidf.get_feature_names()
        list_of_sent = []
        for sent in X[:,0]:
            list_of_sent.append(sent.split())
        temp_vec = self.tf_idf_W2V(feature_names,tf_idf_trans_arr,list_of_sent)
        temp_vec= np.hstack((temp_vec,X[:,[1,2]]))
        return temp_vec
```

```
In [23]: # For simple cv
         #Train data
         X_train = train_df.iloc[:round(train_df.shape[0]*0.70),:]
         X_test_cv = train_df.iloc[round(train_df.shape[0]*0.70):,:]
         #transforming to tfidf weighted word2vec
         tfidfvect_w2v = TfidfWeightedWord2Vec(w2v_model_300)
         tfidfvect_w2v.fit(X_train[['final_text','HelpfulnessNumerator',
                                    'HelpfulnessDenominator']].values)
         X_train_tfw2v = tfidfvect_w2v.transform(X_train[['final_text',
                         'HelpfulnessNumerator','HelpfulnessDenominator']].values)
         X_cv_tfw2v = tfidfvect_w2v.transform(X_test_cv[['final_text',
                          'HelpfulnessNumerator','HelpfulnessDenominator']].values)

In [24]: #scaling the data
         scale = StandardScaler()
         X_train_sc = scale.fit_transform(X_train_tfw2v)
         X_test_cv_sc = scale.transform(X_cv_tfw2v)

In [18]: for i in ParameterGrid({'C':[1,5,7],
                                 'gamma':[0.001,0.008,0.01,0.1,0.5,1,10]}):
             model = SVC(C=i['C'],kernel='rbf',gamma=i['gamma'])
             model.fit(X_train_sc,X_train.Score)
             train_score = model.score(X_train_sc,X_train.Score)
             test_score = model.score(X_test_cv_sc,X_test_cv.Score)
             print('C',i['C'],'Gamma',i['gamma'],'Train Score',train_score,
                              'Test Score',test_score)
```

```
C 1 Gamma 0.001 Train Score 0.9242857142857143 Test Score 0.8833333333333333
C 1 Gamma 0.008 Train Score 0.9641496598639456 Test Score 0.8671428571428571
C 1 Gamma 0.01 Train Score 0.9679591836734693 Test Score 0.8611111111111112
C 1 Gamma 0.1 Train Score 0.9785714285714285 Test Score 0.8268253968253968
C 1 Gamma 0.5 Train Score 0.9789795918367347 Test Score 0.8287301587301588
C 1 Gamma 1 Train Score 0.9792517006802721 Test Score 0.829047619047619
C 1 Gamma 10 Train Score 0.9814285714285714 Test Score 0.8323809523809523
C 5 Gamma 0.001 Train Score 0.9479591836734694 Test Score 0.8898412698412699
C 5 Gamma 0.008 Train Score 0.9782993197278912 Test Score 0.87
C 5 Gamma 0.01 Train Score 0.9786394557823129 Test Score 0.8653968253968254
C 5 Gamma 0.1 Train Score 0.978843537414966 Test Score 0.8274603174603175
C 5 Gamma 0.5 Train Score 0.9793877551020408 Test Score 0.8282539682539682
C 5 Gamma 1 Train Score 0.9798639455782313 Test Score 0.8304761904761905
C 5 Gamma 10 Train Score 0.9821768707482993 Test Score 0.8320634920634921
C 7 Gamma 0.001 Train Score 0.9527210884353742 Test Score 0.8895238095238095
C 7 Gamma 0.008 Train Score 0.9785714285714285 Test Score 0.8701587301587301
C 7 Gamma 0.01 Train Score 0.9785034013605443 Test Score 0.8653968253968254
C 7 Gamma 0.1 Train Score 0.9787755102040816 Test Score 0.8274603174603175
C 7 Gamma 0.5 Train Score 0.9795238095238096 Test Score 0.8284126984126984
C 7 Gamma 1 Train Score 0.980204081632653 Test Score 0.8314285714285714
C 7 Gamma 10 Train Score 0.9823129251700681 Test Score 0.8319047619047619
```

```
In [17]: for i in ParameterGrid({'C':[0.001,0.01,0.1],
                                 'gamma':[0.001,0.008,0.01,0.1,0.5,1,10]}):
             model = SVC(C=i['C'],kernel='rbf',gamma=i['gamma'])
             model.fit(X_train_sc,X_train.Score)
             train_score = model.score(X_train_sc,X_train.Score)
             test_score = model.score(X_test_cv_sc,X_test_cv.Score)
             print('C',i['C'],'Gamma',i['gamma'],'Train Score',train_score,
                              'Test Score',test_score)
```

```
C 0.001 Gamma 0.001 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 0.008 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 0.01 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 0.1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 0.5 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.001 Gamma 10 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 0.001 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 0.008 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 0.01 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 0.1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 0.5 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.01 Gamma 10 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 0.001 Train Score 0.8851020408163265 Test Score 0.8536507936507937
C 0.1 Gamma 0.008 Train Score 0.8639455782312925 Test Score 0.8319047619047619
C 0.1 Gamma 0.01 Train Score 0.8584353741496599 Test Score 0.8273015873015873
C 0.1 Gamma 0.1 Train Score 0.8563945578231292 Test Score 0.8253968253968254
C 0.1 Gamma 0.5 Train Score 0.8571428571428571 Test Score 0.8258730158730159
C 0.1 Gamma 1 Train Score 0.8572789115646259 Test Score 0.8257142857142857
C 0.1 Gamma 10 Train Score 0.8565986394557823 Test Score 0.8257142857142857
```

```
In [20]: c = [10.5,0.85,1,2.5,5,10,12,20]
         gamma = [0.00095,0.001,0.0013,0.0015,0.0024,0.007,0.01,1,10]
         model_grid_tfidfw2v = GridSearchCV(
                                    make_pipeline(TfidfWeightedWord2Vec(w2v_model_300),
                                    StandardScaler(),SVC()),
                                     param_grid={'svc__C': c,'svc__gamma':gamma},
                                    cv=TimeSeriesSplit(n_splits=10),n_jobs=-1)
         model_grid_tfidfw2v.fit(train_df[['final_text','HelpfulnessNumerator',
                                    'HelpfulnessDenominator']].values,train_df.Score)
```

```
In [22]: dict_scores = []
         idx = 0
         for i in model_grid_tfidfw2v.grid_scores_:
             dict_score = []
             dict_score.append(i[0]['svc__gamma'])
```

```
            dict_score.append(i[0]['svc__C'])
            dict_score.append(i[1])
            dict_score.append(i[2].std())
            dict_score.append(model_grid_tfidfw2v.cv_results_['mean_train_score'][idx])
            dict_scores.append(dict_score)
            idx = idx + 1
        scores_df = pd.DataFrame(dict_scores,columns=['gamma','C','Test_score',
                                                      'Test_std','Train_score'])
```

In [25]:
```
scores_df.sort_values('Test_score',ascending=False).head(10)
```

Out[25]:

	gamma	C	Test_score	Test_std	Train_score
64	0.00100	20.0	0.892457	0.004307	0.972534
63	0.00095	20.0	0.892404	0.004359	0.971365
54	0.00095	12.0	0.892142	0.005162	0.964579
47	0.00130	10.0	0.892090	0.005534	0.969587
55	0.00100	12.0	0.892090	0.005008	0.965825
2	0.00130	10.5	0.892038	0.005365	0.970219
56	0.00130	12.0	0.892038	0.005081	0.971907
1	0.00100	10.5	0.891933	0.006247	0.963971
0	0.00095	10.5	0.891776	0.005768	0.962865
45	0.00095	10.0	0.891776	0.006028	0.961995

In [36]:
```
model_random_tfidfw2v = RandomizedSearchCV(
                            make_pipeline(TfidfWeightedWord2Vec(w2v_model_300),
                                StandardScaler(),SVC()),
                            param_distributions={'svc__C': uniform(loc=0,scale=3.5),
                            'svc__gamma':uniform(loc=0.0008,scale=0.004)},n_iter=15,
                                cv=TimeSeriesSplit(n_splits=10),n_jobs=-1)
        model_random_tfidfw2v.fit(train_df[['final_text','HelpfulnessNumerator',
                                    'HelpfulnessDenominator']].values,train_df.Score)
```

In [38]:
```
dict_scores = []
idx = 0
for i in model_random_tfidfw2v.grid_scores_:
    dict_score = []
    dict_score.append(i[0]['svc__gamma'])
    dict_score.append(i[0]['svc__C'])
    dict_score.append(i[1])
    dict_score.append(i[2].std())
    dict_score.append(model_random_tfidfw2v.cv_results_['mean_train_score'][idx])
    dict_scores.append(dict_score)
    idx = idx + 1
scores_df = pd.DataFrame(dict_scores,columns=['gamma','C','Test_score',
                                              'Test_std','Train_score'])
```

In [39]:
```
scores_df.sort_values('Test_score',ascending=False).head(10)
```

Out[39]:

	gamma	C	Test_score	Test_std	Train_score
5	0.002625	2.978150	0.889314	0.006145	0.968132

```
11  0.002274  2.929955    0.889104  0.005996     0.964255
10  0.003434  3.078954    0.887742  0.006286     0.974421
4   0.003403  2.501247    0.887690  0.006506     0.971187
12  0.003382  3.446907    0.887533  0.006390     0.975409
6   0.002573  1.670520    0.887218  0.006713     0.955952
9   0.003608  2.015362    0.886695  0.006492     0.968607
7   0.003702  2.062496    0.886433  0.006320     0.969583
14  0.003818  2.920594    0.886276  0.006151     0.975324
0   0.004238  2.525873    0.885961  0.006509     0.975278
```

best cv score for tfidf word2vec got at gamma = 0.00100 C = 20.0 and mean cv score is 0.892457

In [41]:
```
#testscore
# For simple cv
#transforming to tfidf weighted word2vec
tfidfvect_w2v = TfidfWeightedWord2Vec(w2v_model_300)
tfidfvect_w2v.fit(train_df[['final_text','HelpfulnessNumerator',
                            'HelpfulnessDenominator']].values)
X_train_tfw2v = tfidfvect_w2v.transform(train_df[['final_text',
                'HelpfulnessNumerator','HelpfulnessDenominator']].values)
X_cv_tfw2v = tfidfvect_w2v.transform(test_df[['final_text',
                 'HelpfulnessNumerator','HelpfulnessDenominator']].values)

#scaling the data
scale = StandardScaler()
X_train_sc = scale.fit_transform(X_train_tfw2v)
X_test_cv_sc = scale.transform(X_cv_tfw2v)

model = SVC(C=20,kernel='rbf',gamma=0.00100)
model.fit(X_train_sc,train_df.Score)
#Predicting training data
train_list = model.predict(X_train_sc)
#Accuracy score
score_train = accuracy_score(train_df.Score,train_list)
#predict test cv
test_list = model.predict(X_test_cv_sc)
#Accuracy score
score_test = accuracy_score(test_df.Score,test_list)
#precision
#precision
test_precision = precision_score(test_df.Score,test_list)
#recall
test_recall = recall_score(test_df.Score,test_list)
#confusion matrix
confusion_matrix_test = confusion_matrix(test_df.Score,test_list)
print('C' ,20,'gamma',0.00100)
print('Train Score', score_train)
print('Test Score',score_test)
```

```
        print('Test Precision',test_precision)
        print('Test Recall',test_recall)
        print('Test ConfusionMatrix',confusion_matrix_test)
        print('No of support vectors for each class',model.n_support_)
```

```
C 20 gamma 0.001
Train Score 0.9643809523809523
Test Score 0.8904444444444445
Test Precision 0.8969719350073855
Test Recall 0.9799623453469607
Test ConfusionMatrix [[ 727  837]
 [ 149 7287]]
No of support vectors for each class [2031 2750]
```

SGD Classifier

```
In [58]: for i in ParameterGrid({'alpha':[0.00005,0.00008,0.0001,0.00012],
                                'l1_ratio':[0,0.03,0.05,0.08,0.1,0.15,0.25,0.35,
                                            0.45,0.55,0.65,0.75,0.85,0.95],
                                'penality':['l1','l2','elasticnet']}):
            model = SGDClassifier(penalty=i['penality'],alpha=i['alpha'],
                                  l1_ratio=i['l1_ratio'],random_state=25)
            model.fit(X_train_sc,X_train.Score)
            train_score = model.score(X_train_sc,X_train.Score)
            test_score = model.score(X_test_cv_sc,X_test_cv.Score)
            print('Alpha',i['alpha'],'l1_ratio',i['l1_ratio'],'Penality',i['penality'],
                  'Train Score',train_score,'Test Score',test_score)
```

```
Alpha 5e-05 l1_ratio 0 Penality l1 Train Score 0.9236734693877551 Test Score 0.892222222222222
Alpha 5e-05 l1_ratio 0 Penality l2 Train Score 0.9210884353741496 Test Score 0.889047619047619
Alpha 5e-05 l1_ratio 0 Penality elasticnet Train Score 0.9210884353741496 Test Score 0.8890476
Alpha 5e-05 l1_ratio 0.03 Penality l1 Train Score 0.9236734693877551 Test Score 0.892222222222
Alpha 5e-05 l1_ratio 0.03 Penality l2 Train Score 0.9210884353741496 Test Score 0.889047619047
Alpha 5e-05 l1_ratio 0.03 Penality elasticnet Train Score 0.9169387755102041 Test Score 0.8917
Alpha 5e-05 l1_ratio 0.05 Penality l1 Train Score 0.9236734693877551 Test Score 0.892222222222
Alpha 5e-05 l1_ratio 0.05 Penality l2 Train Score 0.9210884353741496 Test Score 0.889047619047
Alpha 5e-05 l1_ratio 0.05 Penality elasticnet Train Score 0.9196598639455782 Test Score 0.8933
Alpha 5e-05 l1_ratio 0.08 Penality l1 Train Score 0.9236734693877551 Test Score 0.892222222222
Alpha 5e-05 l1_ratio 0.08 Penality l2 Train Score 0.9210884353741496 Test Score 0.889047619047
Alpha 5e-05 l1_ratio 0.08 Penality elasticnet Train Score 0.9186394557823129 Test Score 0.8884
Alpha 5e-05 l1_ratio 0.1 Penality l1 Train Score 0.9236734693877551 Test Score 0.892222222222
Alpha 5e-05 l1_ratio 0.1 Penality l2 Train Score 0.9210884353741496 Test Score 0.889047619047
Alpha 5e-05 l1_ratio 0.1 Penality elasticnet Train Score 0.9102040816326531 Test Score 0.8865
Alpha 5e-05 l1_ratio 0.15 Penality l1 Train Score 0.9236734693877551 Test Score 0.892222222222
Alpha 5e-05 l1_ratio 0.15 Penality l2 Train Score 0.9210884353741496 Test Score 0.889047619047
Alpha 5e-05 l1_ratio 0.15 Penality elasticnet Train Score 0.9124489795918367 Test Score 0.8831
Alpha 5e-05 l1_ratio 0.25 Penality l1 Train Score 0.9236734693877551 Test Score 0.892222222222
```

```
Alpha 5e-05 l1_ratio 0.25 Penality l2 Train Score 0.9210884353741496 Test Score 0.889047619047
Alpha 5e-05 l1_ratio 0.25 Penality elasticnet Train Score 0.9208163265306123 Test Score 0.89238
Alpha 5e-05 l1_ratio 0.35 Penality l1 Train Score 0.9236734693877551 Test Score 0.892222222222
Alpha 5e-05 l1_ratio 0.35 Penality l2 Train Score 0.9210884353741496 Test Score 0.889047619047
Alpha 5e-05 l1_ratio 0.35 Penality elasticnet Train Score 0.9231972789115647 Test Score 0.8953
Alpha 5e-05 l1_ratio 0.45 Penality l1 Train Score 0.9236734693877551 Test Score 0.892222222222
Alpha 5e-05 l1_ratio 0.45 Penality l2 Train Score 0.9210884353741496 Test Score 0.889047619047
Alpha 5e-05 l1_ratio 0.45 Penality elasticnet Train Score 0.922312925170068 Test Score 0.89333
Alpha 5e-05 l1_ratio 0.55 Penality l1 Train Score 0.9236734693877551 Test Score 0.892222222222
Alpha 5e-05 l1_ratio 0.55 Penality l2 Train Score 0.9210884353741496 Test Score 0.889047619047
Alpha 5e-05 l1_ratio 0.55 Penality elasticnet Train Score 0.9128571428571428 Test Score 0.8839
Alpha 5e-05 l1_ratio 0.65 Penality l1 Train Score 0.9236734693877551 Test Score 0.892222222222
Alpha 5e-05 l1_ratio 0.65 Penality l2 Train Score 0.9210884353741496 Test Score 0.889047619047
Alpha 5e-05 l1_ratio 0.65 Penality elasticnet Train Score 0.9136734693877551 Test Score 0.8855
Alpha 5e-05 l1_ratio 0.75 Penality l1 Train Score 0.9236734693877551 Test Score 0.892222222222
Alpha 5e-05 l1_ratio 0.75 Penality l2 Train Score 0.9210884353741496 Test Score 0.889047619047
Alpha 5e-05 l1_ratio 0.75 Penality elasticnet Train Score 0.9219727891156463 Test Score 0.8926
Alpha 5e-05 l1_ratio 0.85 Penality l1 Train Score 0.9236734693877551 Test Score 0.892222222222
Alpha 5e-05 l1_ratio 0.85 Penality l2 Train Score 0.9210884353741496 Test Score 0.889047619047
Alpha 5e-05 l1_ratio 0.85 Penality elasticnet Train Score 0.9263945578231293 Test Score 0.8974
Alpha 5e-05 l1_ratio 0.95 Penality l1 Train Score 0.9236734693877551 Test Score 0.892222222222
Alpha 5e-05 l1_ratio 0.95 Penality l2 Train Score 0.9210884353741496 Test Score 0.889047619047
Alpha 5e-05 l1_ratio 0.95 Penality elasticnet Train Score 0.9259863945578232 Test Score 0.8965
Alpha 8e-05 l1_ratio 0 Penality l1 Train Score 0.9240136054421769 Test Score 0.894920634920634
Alpha 8e-05 l1_ratio 0 Penality l2 Train Score 0.9206122448979592 Test Score 0.89206349206349
Alpha 8e-05 l1_ratio 0 Penality elasticnet Train Score 0.9206122448979592 Test Score 0.8920634
Alpha 8e-05 l1_ratio 0.03 Penality l1 Train Score 0.9240136054421769 Test Score 0.894920634920
Alpha 8e-05 l1_ratio 0.03 Penality l2 Train Score 0.9206122448979592 Test Score 0.892063492063
Alpha 8e-05 l1_ratio 0.03 Penality elasticnet Train Score 0.9195238095238095 Test Score 0.8931
Alpha 8e-05 l1_ratio 0.05 Penality l1 Train Score 0.9240136054421769 Test Score 0.894920634920
Alpha 8e-05 l1_ratio 0.05 Penality l2 Train Score 0.9206122448979592 Test Score 0.892063492063
Alpha 8e-05 l1_ratio 0.05 Penality elasticnet Train Score 0.9224489795918367 Test Score 0.8944
Alpha 8e-05 l1_ratio 0.08 Penality l1 Train Score 0.9240136054421769 Test Score 0.894920634920
Alpha 8e-05 l1_ratio 0.08 Penality l2 Train Score 0.9206122448979592 Test Score 0.892063492063
Alpha 8e-05 l1_ratio 0.08 Penality elasticnet Train Score 0.9149659863945578 Test Score 0.8860
Alpha 8e-05 l1_ratio 0.1 Penality l1 Train Score 0.9240136054421769 Test Score 0.89492063492063
Alpha 8e-05 l1_ratio 0.1 Penality l2 Train Score 0.9206122448979592 Test Score 0.89206349206349
Alpha 8e-05 l1_ratio 0.1 Penality elasticnet Train Score 0.9220408163265306 Test Score 0.89301
Alpha 8e-05 l1_ratio 0.15 Penality l1 Train Score 0.9240136054421769 Test Score 0.894920634920
Alpha 8e-05 l1_ratio 0.15 Penality l2 Train Score 0.9206122448979592 Test Score 0.892063492063
Alpha 8e-05 l1_ratio 0.15 Penality elasticnet Train Score 0.9148979591836734 Test Score 0.8853
Alpha 8e-05 l1_ratio 0.25 Penality l1 Train Score 0.9240136054421769 Test Score 0.894920634920
Alpha 8e-05 l1_ratio 0.25 Penality l2 Train Score 0.9206122448979592 Test Score 0.892063492063
Alpha 8e-05 l1_ratio 0.25 Penality elasticnet Train Score 0.9210884353741496 Test Score 0.8936
Alpha 8e-05 l1_ratio 0.35 Penality l1 Train Score 0.9240136054421769 Test Score 0.894920634920
Alpha 8e-05 l1_ratio 0.35 Penality l2 Train Score 0.9206122448979592 Test Score 0.892063492063
Alpha 8e-05 l1_ratio 0.35 Penality elasticnet Train Score 0.9227210884353741 Test Score 0.8946
Alpha 8e-05 l1_ratio 0.45 Penality l1 Train Score 0.9240136054421769 Test Score 0.894920634920
```

```
Alpha 8e-05 l1_ratio 0.45 Penality l2 Train Score 0.9206122448979592 Test Score 0.892063492063
Alpha 8e-05 l1_ratio 0.45 Penality elasticnet Train Score 0.9160544217687074 Test Score 0.88968
Alpha 8e-05 l1_ratio 0.55 Penality l1 Train Score 0.9240136054421769 Test Score 0.894920634920
Alpha 8e-05 l1_ratio 0.55 Penality l2 Train Score 0.9206122448979592 Test Score 0.892063492063
Alpha 8e-05 l1_ratio 0.55 Penality elasticnet Train Score 0.9230612244897959 Test Score 0.8961
Alpha 8e-05 l1_ratio 0.65 Penality l1 Train Score 0.9240136054421769 Test Score 0.894920634920
Alpha 8e-05 l1_ratio 0.65 Penality l2 Train Score 0.9206122448979592 Test Score 0.892063492063
Alpha 8e-05 l1_ratio 0.65 Penality elasticnet Train Score 0.9244897959183673 Test Score 0.8957
Alpha 8e-05 l1_ratio 0.75 Penality l1 Train Score 0.9240136054421769 Test Score 0.894920634920
Alpha 8e-05 l1_ratio 0.75 Penality l2 Train Score 0.9206122448979592 Test Score 0.892063492063
Alpha 8e-05 l1_ratio 0.75 Penality elasticnet Train Score 0.925578231292517 Test Score 0.89428
Alpha 8e-05 l1_ratio 0.85 Penality l1 Train Score 0.9240136054421769 Test Score 0.894920634920
Alpha 8e-05 l1_ratio 0.85 Penality l2 Train Score 0.9206122448979592 Test Score 0.892063492063
Alpha 8e-05 l1_ratio 0.85 Penality elasticnet Train Score 0.923265306122449 Test Score 0.89682
Alpha 8e-05 l1_ratio 0.95 Penality l1 Train Score 0.9240136054421769 Test Score 0.894920634920
Alpha 8e-05 l1_ratio 0.95 Penality l2 Train Score 0.9206122448979592 Test Score 0.892063492063
Alpha 8e-05 l1_ratio 0.95 Penality elasticnet Train Score 0.9178231292517007 Test Score 0.8874
Alpha 0.0001 l1_ratio 0 Penality l1 Train Score 0.9182312925170067 Test Score 0.89
Alpha 0.0001 l1_ratio 0 Penality l2 Train Score 0.9231292517006803 Test Score 0.893968253968254
Alpha 0.0001 l1_ratio 0 Penality elasticnet Train Score 0.9231292517006803 Test Score 0.893968
Alpha 0.0001 l1_ratio 0.03 Penality l1 Train Score 0.9182312925170067 Test Score 0.89
Alpha 0.0001 l1_ratio 0.03 Penality l2 Train Score 0.9231292517006803 Test Score 0.89396825396
Alpha 0.0001 l1_ratio 0.03 Penality elasticnet Train Score 0.921156462585034 Test Score 0.8931
Alpha 0.0001 l1_ratio 0.05 Penality l1 Train Score 0.9182312925170067 Test Score 0.89
Alpha 0.0001 l1_ratio 0.05 Penality l2 Train Score 0.9231292517006803 Test Score 0.89396825396
Alpha 0.0001 l1_ratio 0.05 Penality elasticnet Train Score 0.921156462585034 Test Score 0.8928
Alpha 0.0001 l1_ratio 0.08 Penality l1 Train Score 0.9182312925170067 Test Score 0.89
Alpha 0.0001 l1_ratio 0.08 Penality l2 Train Score 0.9231292517006803 Test Score 0.89396825396
Alpha 0.0001 l1_ratio 0.08 Penality elasticnet Train Score 0.9155102040816326 Test Score 0.886
Alpha 0.0001 l1_ratio 0.1 Penality l1 Train Score 0.9182312925170067 Test Score 0.89
Alpha 0.0001 l1_ratio 0.1 Penality l2 Train Score 0.9231292517006803 Test Score 0.893968253968
Alpha 0.0001 l1_ratio 0.1 Penality elasticnet Train Score 0.9214965986394558 Test Score 0.8934
Alpha 0.0001 l1_ratio 0.15 Penality l1 Train Score 0.9182312925170067 Test Score 0.89
Alpha 0.0001 l1_ratio 0.15 Penality l2 Train Score 0.9231292517006803 Test Score 0.89396825396
Alpha 0.0001 l1_ratio 0.15 Penality elasticnet Train Score 0.9210884353741496 Test Score 0.893
Alpha 0.0001 l1_ratio 0.25 Penality l1 Train Score 0.9182312925170067 Test Score 0.89
Alpha 0.0001 l1_ratio 0.25 Penality l2 Train Score 0.9231292517006803 Test Score 0.89396825396
Alpha 0.0001 l1_ratio 0.25 Penality elasticnet Train Score 0.9206122448979592 Test Score 0.895
Alpha 0.0001 l1_ratio 0.35 Penality l1 Train Score 0.9182312925170067 Test Score 0.89
Alpha 0.0001 l1_ratio 0.35 Penality l2 Train Score 0.9231292517006803 Test Score 0.89396825396
Alpha 0.0001 l1_ratio 0.35 Penality elasticnet Train Score 0.9197278911564626 Test Score 0.888
Alpha 0.0001 l1_ratio 0.45 Penality l1 Train Score 0.9182312925170067 Test Score 0.89
Alpha 0.0001 l1_ratio 0.45 Penality l2 Train Score 0.9231292517006803 Test Score 0.89396825396
Alpha 0.0001 l1_ratio 0.45 Penality elasticnet Train Score 0.9230612244897959 Test Score 0.893
Alpha 0.0001 l1_ratio 0.55 Penality l1 Train Score 0.9182312925170067 Test Score 0.89
Alpha 0.0001 l1_ratio 0.55 Penality l2 Train Score 0.9231292517006803 Test Score 0.89396825396
Alpha 0.0001 l1_ratio 0.55 Penality elasticnet Train Score 0.9242857142857143 Test Score 0.895
Alpha 0.0001 l1_ratio 0.65 Penality l1 Train Score 0.9182312925170067 Test Score 0.89
```

```
Alpha 0.0001 l1_ratio 0.65 Penality l2 Train Score 0.9231292517006803 Test Score 0.893968253968
Alpha 0.0001 l1_ratio 0.65 Penality elasticnet Train Score 0.92421768707483 Test Score 0.89492
Alpha 0.0001 l1_ratio 0.75 Penality l1 Train Score 0.9182312925170067 Test Score 0.89
Alpha 0.0001 l1_ratio 0.75 Penality l2 Train Score 0.9231292517006803 Test Score 0.893968253968
Alpha 0.0001 l1_ratio 0.75 Penality elasticnet Train Score 0.9238095238095239 Test Score 0.895
Alpha 0.0001 l1_ratio 0.85 Penality l1 Train Score 0.9182312925170067 Test Score 0.89
Alpha 0.0001 l1_ratio 0.85 Penality l2 Train Score 0.9231292517006803 Test Score 0.893968253968
Alpha 0.0001 l1_ratio 0.85 Penality elasticnet Train Score 0.9238775510204081 Test Score 0.895
Alpha 0.0001 l1_ratio 0.95 Penality l1 Train Score 0.9182312925170067 Test Score 0.89
Alpha 0.0001 l1_ratio 0.95 Penality l2 Train Score 0.9231292517006803 Test Score 0.893968253968
Alpha 0.0001 l1_ratio 0.95 Penality elasticnet Train Score 0.9229251700680272 Test Score 0.895
Alpha 0.00012 l1_ratio 0 Penality l1 Train Score 0.9231972789115647 Test Score 0.894603174603
Alpha 0.00012 l1_ratio 0 Penality l2 Train Score 0.9166666666666666 Test Score 0.888412698412
Alpha 0.00012 l1_ratio 0 Penality elasticnet Train Score 0.9166666666666666 Test Score 0.88841
Alpha 0.00012 l1_ratio 0.03 Penality l1 Train Score 0.9231972789115647 Test Score 0.894603174
Alpha 0.00012 l1_ratio 0.03 Penality l2 Train Score 0.9166666666666666 Test Score 0.888412698
Alpha 0.00012 l1_ratio 0.03 Penality elasticnet Train Score 0.9204761904761904 Test Score 0.89
Alpha 0.00012 l1_ratio 0.05 Penality l1 Train Score 0.9231972789115647 Test Score 0.894603174
Alpha 0.00012 l1_ratio 0.05 Penality l2 Train Score 0.9166666666666666 Test Score 0.888412698
Alpha 0.00012 l1_ratio 0.05 Penality elasticnet Train Score 0.9219727891156463 Test Score 0.89
Alpha 0.00012 l1_ratio 0.08 Penality l1 Train Score 0.9231972789115647 Test Score 0.894603174
Alpha 0.00012 l1_ratio 0.08 Penality l2 Train Score 0.9166666666666666 Test Score 0.888412698
Alpha 0.00012 l1_ratio 0.08 Penality elasticnet Train Score 0.9227210884353741 Test Score 0.89
Alpha 0.00012 l1_ratio 0.1 Penality l1 Train Score 0.9231972789115647 Test Score 0.89460317460
Alpha 0.00012 l1_ratio 0.1 Penality l2 Train Score 0.9166666666666666 Test Score 0.88841269841
Alpha 0.00012 l1_ratio 0.1 Penality elasticnet Train Score 0.9228571428571428 Test Score 0.894
Alpha 0.00012 l1_ratio 0.15 Penality l1 Train Score 0.9231972789115647 Test Score 0.894603174
Alpha 0.00012 l1_ratio 0.15 Penality l2 Train Score 0.9166666666666666 Test Score 0.888412698
Alpha 0.00012 l1_ratio 0.15 Penality elasticnet Train Score 0.9222448979591836 Test Score 0.89
Alpha 0.00012 l1_ratio 0.25 Penality l1 Train Score 0.9231972789115647 Test Score 0.894603174
Alpha 0.00012 l1_ratio 0.25 Penality l2 Train Score 0.9166666666666666 Test Score 0.888412698
Alpha 0.00012 l1_ratio 0.25 Penality elasticnet Train Score 0.921156462585034 Test Score 0.893
Alpha 0.00012 l1_ratio 0.35 Penality l1 Train Score 0.9231972789115647 Test Score 0.894603174
Alpha 0.00012 l1_ratio 0.35 Penality l2 Train Score 0.9166666666666666 Test Score 0.888412698
Alpha 0.00012 l1_ratio 0.35 Penality elasticnet Train Score 0.924625850340136 Test Score 0.893
Alpha 0.00012 l1_ratio 0.45 Penality l1 Train Score 0.9231972789115647 Test Score 0.894603174
Alpha 0.00012 l1_ratio 0.45 Penality l2 Train Score 0.9166666666666666 Test Score 0.888412698
Alpha 0.00012 l1_ratio 0.45 Penality elasticnet Train Score 0.9231292517006803 Test Score 0.894
Alpha 0.00012 l1_ratio 0.55 Penality l1 Train Score 0.9231972789115647 Test Score 0.894603174
Alpha 0.00012 l1_ratio 0.55 Penality l2 Train Score 0.9166666666666666 Test Score 0.888412698
Alpha 0.00012 l1_ratio 0.55 Penality elasticnet Train Score 0.9229931972789116 Test Score 0.894
Alpha 0.00012 l1_ratio 0.65 Penality l1 Train Score 0.9231972789115647 Test Score 0.894603174
Alpha 0.00012 l1_ratio 0.65 Penality l2 Train Score 0.9166666666666666 Test Score 0.888412698
Alpha 0.00012 l1_ratio 0.65 Penality elasticnet Train Score 0.9238775510204081 Test Score 0.896
Alpha 0.00012 l1_ratio 0.75 Penality l1 Train Score 0.9231972789115647 Test Score 0.894603174
Alpha 0.00012 l1_ratio 0.75 Penality l2 Train Score 0.9166666666666666 Test Score 0.888412698
Alpha 0.00012 l1_ratio 0.75 Penality elasticnet Train Score 0.9242857142857143 Test Score 0.894
Alpha 0.00012 l1_ratio 0.85 Penality l1 Train Score 0.9231972789115647 Test Score 0.894603174
```

```
Alpha 0.00012 l1_ratio 0.85 Penality l2 Train Score 0.9166666666666666 Test Score 0.8884126984
Alpha 0.00012 l1_ratio 0.85 Penality elasticnet Train Score 0.9204081632653062 Test Score 0.891
Alpha 0.00012 l1_ratio 0.95 Penality l1 Train Score 0.9231972789115647 Test Score 0.8946031746
Alpha 0.00012 l1_ratio 0.95 Penality l2 Train Score 0.9166666666666666 Test Score 0.8884126984
Alpha 0.00012 l1_ratio 0.95 Penality elasticnet Train Score 0.9236054421768708 Test Score 0.895
```

```
In [25]: model_random_tfidfw2v = RandomizedSearchCV(
                              make_pipeline(TfidfWeightedWord2Vec(w2v_model_300),
                              StandardScaler(),SGDClassifier(n_jobs=-1)),
                              param_distributions={'sgdclassifier__penalty':['l1','l2'],
                              'sgdclassifier__alpha':uniform(loc=0.00001,scale=0.0049),
                              'sgdclassifier__l1_ratio':uniform(loc=0,scale=1)},n_iter=40,
                                cv=TimeSeriesSplit(n_splits=10),n_jobs=-1)
         model_random_tfidfw2v.fit(train_df[['final_text','HelpfulnessNumerator',
                                        'HelpfulnessDenominator']].values,train_df.Score)
```

```
In [27]: dict_scores = []
         idx = 0
         for i in model_random_tfidfw2v.grid_scores_:
             dict_score = []
             dict_score.append(i[0]['sgdclassifier__alpha'])
             dict_score.append(i[0]['sgdclassifier__l1_ratio'])
             dict_score.append(i[0]['sgdclassifier__penalty'])
             dict_score.append(i[1])
             dict_score.append(i[2].std())
             dict_score.append(model_random_tfidfw2v.cv_results_['mean_train_score'][idx])
             dict_scores.append(dict_score)
             idx = idx + 1
         scores_df = pd.DataFrame(dict_scores,columns=['alpha','l1_ratio','penalty','Test_score
                                                       'Test_std','Train_score'])
```

```
In [29]: scores_df.sort_values('Test_score',ascending=False).head(10)
```

Out[29]:

	alpha	l1_ratio	penalty	Test_score	Test_std	Train_score
25	0.000823	0.464483	l1	0.892981	0.008003	0.924661
16	0.000456	0.864083	l1	0.892823	0.007788	0.926553
29	0.000856	0.372678	l1	0.891881	0.008156	0.924244
36	0.000393	0.995435	l1	0.891828	0.007841	0.928939
31	0.000906	0.480030	l1	0.891671	0.005621	0.924880
37	0.000734	0.360432	l1	0.891566	0.007266	0.926045
15	0.000525	0.521241	l1	0.891409	0.008473	0.927267
19	0.000608	0.692168	l1	0.891252	0.007907	0.926134
4	0.000154	0.842079	l2	0.891200	0.007756	0.925755
14	0.000231	0.884751	l2	0.890676	0.006560	0.924619

Got best cv scores at alpha = 0.000823 l1_ratio = 0.464483 penalty = l1 and mean cv score is 0.892981.

In [34]:
```
#testscore
#transforming to tfidf weighted word2vec
tfidfvect_w2v = TfidfWeightedWord2Vec(w2v_model_300)
tfidfvect_w2v.fit(train_df[['final_text','HelpfulnessNumerator',
                            'HelpfulnessDenominator']].values)
X_train_tfw2v = tfidfvect_w2v.transform(train_df[['final_text',
                'HelpfulnessNumerator','HelpfulnessDenominator']].values)
X_cv_tfw2v = tfidfvect_w2v.transform(test_df[['final_text',
                 'HelpfulnessNumerator','HelpfulnessDenominator']].values)

#scaling the data
scale = StandardScaler()
X_train_sc = scale.fit_transform(X_train_tfw2v)
X_test_cv_sc = scale.transform(X_cv_tfw2v)

model = SGDClassifier(penalty='l1',alpha=0.000823,l1_ratio=0.464483,random_state=25)
model.fit(X_train_sc,train_df.Score)
#Predicting training data
train_list = model.predict(X_train_sc)
#Accuracy score
score_train = accuracy_score(train_df.Score,train_list)
#predict test cv
test_list = model.predict(X_test_cv_sc)
#Accuracy score
score_test = accuracy_score(test_df.Score,test_list)
#precision
#precision
test_precision = precision_score(test_df.Score,test_list)
#recall
test_recall = recall_score(test_df.Score,test_list)
#confusion matrix
confusion_matrix_test = confusion_matrix(test_df.Score,test_list)
print("penalty='l1',alpha=0.000823,l1_ratio=0.464483")
print('Train Score', score_train)
print('Test Score',score_test)
print('Test Precision',test_precision)
print('Test Recall',test_recall)
print('Test ConfusionMatrix',confusion_matrix_test)
```

```
penalty='l1',alpha=0.000823,l1_ratio=0.464483
Train Score 0.9137142857142857
Test Score 0.889
Test Precision 0.9022622172228472
Test Recall 0.9708176438945669
Test ConfusionMatrix [[ 782  782]
 [ 217 7219]]
```

Observations:

1. For Binary Bag of Words got high mean cv at `gamma = 0.010 ,C = 7.0` and cv mean score is `0.921215`
    - Train Score `0.9915238095238095`
    - Test Score `0.9246666666666666`
    - Test Precision `0.9447222953408791`
    - Test Recall `0.9653039268423884`
    - No of support vectors for each class `[2245, 3591]`
    - Test Confusion Matrix

$$\begin{bmatrix} 1144 & 420 \\ 258 & 7287 \end{bmatrix} \tag{1}$$

2. SGD: For Binary bagof words wit SGD got best cv score at `alpha = 0.003570, l1_ratio = 0.912537, penalty l2` and corresponding mean cv test score is `0.915977`
    - Train Score `0.9445238095238095`
    - Test Score `0.9197777777777778`
    - Test Precision `0.930053804765565`
    - Test Recall `0.9763313609467456`
    - Test Confusion Matrix

$$\begin{bmatrix} 1018 & 546 \\ 176 & 7260 \end{bmatrix} \tag{2}$$

3. For Tf-Idf got high mean cv at `gamma = 0.175194, C = 7.107109` and mean cv is `0.924987`
    - Train Score `1.0`
    - Test Score `0.9415555555555556`
    - Test Precision `0.9508089770354906`
    - Test Recall `0.9799623453469607`
    - No of support vectors for each class `[2810 6720]`
    - Test Confusion Matrix

$$\begin{bmatrix} 1187 & 377 \\ 149 & 7287 \end{bmatrix} \tag{3}$$

4. SGD: For Tf-Idf with sgd got best scores at `alpha = 0.00003, l1_ratio = 0.15, penalty = l1` and mean cv score is `0.929282`
    - Train Score `0.9569047619047619`
    - Test Score `0.9355555555555556`
    - Test Precision `0.9517659462308908`
    - Test Recall `0.9712210866057019`
    - Test Confusion Matrix

$$\begin{bmatrix} 1198 & 366 \\ 214 & 7222 \end{bmatrix} \tag{4}$$

5. For Avg Word2Vec got high mean cv at `gamma = 0.00100, C = 10` and mean cv score is `0.933630`

- Train Score 0.9725714285714285
- Test Score 0.9332222222222222
- Test Precision 0.9459741615555266
- Test Recall 0.9748520710059172
- No of support vectors for each class [1693 2078]

$$\begin{bmatrix} 1150 & 414 \\ 187 & 7249 \end{bmatrix} \tag{5}$$

6. SGD: For Avg Word2Vec got best cv mean score at alpha = 0.00012,l1_ratio= 0.25 penlty = l1 and mean cv score is 0.925406

   - Train Score 0.9392380952380952
   - Test Score 0.9287777777777778
   - Test Precision 0.9474516001580403
   - Test Recall 0.9674556213017751
   - Test ConfusionMatrix

$$\begin{bmatrix} 1165 & 399 \\ 242 & 7194 \end{bmatrix} \tag{6}$$

7. For Tf-Idf Word2Vec got high mean cv at gamma = 0.00100 C = 20.0 and mean cv score is 0.892457

   - Train Score 0.9643809523809523
   - Test Score 0.8904444444444445
   - Test Precision 0.8969719350073855
   - Test Recall 0.9799623453469607
   - No of support vectors for each class [2031 2750]

$$\begin{bmatrix} 727 & 837 \\ 149 & 7287 \end{bmatrix} \tag{7}$$

8. SGD : Got best cv scores at alpha = 0.000823 l1_ratio = 0.464483 penalty = l1 and mean cv score is 0.892981.

   - Train Score 0.9137142857142857
   - Test Score 0.889
   - Test Precision 0.9022622172228472
   - Test Recall 0.9708176438945669
   - Test ConfusionMatrix

$$\begin{bmatrix} 782 & 782 \\ 217 & 7219 \end{bmatrix} \tag{8}$$