logistic_regression.py

November 9, 2019

0.1 Logistic Regression Model

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

In [9]:
```
# read the first 500,000 yelp reviews
# df = pd.read_json('yelp_dataset/review.json', lines = True)
# df = df[0:500000]
df = pd.read_csv("yelp_dataset/yelp_reviews.csv", encoding='utf-8')
```

In [9]:
```
df.head(5)
```

Out[9]:

	business_id	cool	date	funny	review_id	stars	text	useful	user_id
0	ujmEBvifdJM6h6RLv4wQIg	0	2013-05-07 04:34:36	1	Q1sbwvVQXV2734tPgoKj4Q	1	Total bill for this horrible service? Over $8G...	6	
1	NZnhc2sEQy3RmzKTZnqtwQ	0	2017-01-14 21:30:33	0	GJXCdrto3ASJOqKeVWPi6Q	5	I *adore* Travis at the Hard Rock's new Kelly ...	0	
2	WTqjgwHlXbSFevF32_DJVw	0	2016-11-09 20:09:03	0	2TzJjDVDEuAW6MR5Vuc1ug	5	I have to say that this office really has it t...	3	
3	ikCg8xy5JIg_NGPx-MSIDA	0	2018-01-09 20:56:38	0	yi0R0Ugj_xUx_Nek0-_Qig	5	Went in for a lunch. Steak sandwich was delici...	0	
4	b1b1eb3uo-w561D0ZfCEiQ	0	2018-01-30 23:07:38	0	11a8sVPMUFtaC7_ABRkmtw	1	Today was my second out of three sessions I ha...	7	

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

In [7]:
```
df.describe()
```

Out[7]:

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

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500000 entries, 0 to 499999
Data columns (total 9 columns):
business_id    500000 non-null object
cool           500000 non-null int64
date           500000 non-null datetime64[ns]
funny          500000 non-null int64
review_id      500000 non-null object
stars          500000 non-null int64
text           500000 non-null object
useful         500000 non-null int64
user_id        500000 non-null object
dtypes: datetime64[ns](1), int64(4), object(4)
memory usage: 34.3+ MB
```

In [10]:
```
# split the dataset into training and validation datasets
train_x, valid_x, train_y, valid_y = train_test_split(df['text'], df['stars'])
```

In [19]:
```
# TRAIN THE MODEL AND CALCULATE PERFORMANCE METRICS (ACCURACY, PRECISION, RECALL, F-S
# FOR BOTH TRAINING AND TEST SET
# Weighted performance metrics
def train_model_weighted(classifier, feature_vector_train, label, feature_vector_vali
    # fit the training dataset on the classifier
    classifier.fit(feature_vector_train, label)

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

```
    test_predictions = classifier.predict(feature_vector_valid)

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

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

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

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

In [34]:
```
# word level tf-idf
tfidf_vect = TfidfVectorizer(lowercase = True, analyzer='word', token_pattern=r'[a-zA-
                             max_features=500)
tfidf_vect.fit(df['text'])
xtrain_tfidf =  tfidf_vect.transform(train_x)
xvalid_tfidf =  tfidf_vect.transform(valid_x)
```

0.2 Model 1 : Bag of word representation - Word level

In [35]:
```
# Linear Classifier on Word Level TF IDF Vectors
# C (penalty) : 1 (Default)
# Solver - Liblinear (Default)
# Multiclass - OVR (one versus rest)
# Default for max_iter is 100 which means that
# the solver either coverges within 100 iteration or stops after 100 iterations
results = train_model_weighted(LogisticRegression(), xtrain_tfidf, train_y, xvalid_tfi
print ("LR, WordLevel TF-IDF train accuracy: ", results[1][0])
print("")
print ("LR, WordLevel TF-IDF train precision: ", results[1][1])
print("")
print ("LR, WordLevel TF-IDF train recall: ", results[1][2])
print("")
print ("LR, WordLevel TF-IDF train f1_score: ", results[1][3])
print("**********************************************************")
print ("LR, WordLevel TF-IDF test accuracy: ", results[0][0])
print("")
print ("LR, WordLevel TF-IDF test precision: ", results[0][1])
print("")
```

```
        print ("LR, WordLevel TF-IDF test recall: ", results[0][2])
        print("")
        print ("LR, WordLevel TF-IDF test f1_score: ", results[0][3])
```

```
LR, WordLevel TF-IDF train accuracy:  0.441184

LR, WordLevel TF-IDF train precision:  0.194643321856

LR, WordLevel TF-IDF train recall:  0.441184

LR, WordLevel TF-IDF train f1_score:  0.2701158517663255
*******************************************************
LR, WordLevel TF-IDF test accuracy:  0.439264

LR, WordLevel TF-IDF test precision:  0.192952861696

LR, WordLevel TF-IDF test recall:  0.439264

LR, WordLevel TF-IDF test f1_score:  0.2681271284434266
```

0.3 Model : Bag of words representation - word level - 10 fold Cross Validation

In [39]:
```
# 10 Fold cross validation
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100],
              'solver': ["newton-cg", "lbfgs", "liblinear"],
             'fit_intercept': [True, False]}

logregtf = LogisticRegression(multi_class = "auto")
logreg_cv_tf = GridSearchCV(logregtf, param_grid, cv=10)
logreg_cv_tf.fit(xtrain_tfidf,train_y)
logreg_cv_tf.score(xvalid_tfidf, valid_y)
```

```
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWa
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWa
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWa
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWa
```

```
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWar
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWar
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWar
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWar
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWar
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWar
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWar
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWar
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWar
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWar
```

```
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWa
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWa
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWa
  warnings.warn('Line Search failed')
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/scipy/optimize/linesearch.py:313: Line
  warn('The line search algorithm did not converge', LineSearchWarning)
/Users/anjaliverma/anaconda3/lib/python3.6/site-packages/sklearn/utils/optimize.py:195: UserWa
  warnings.warn('Line Search failed')
```

```
Out[39]: 0.439264
```

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

```
In [12]: # ngram level tf-idf
         tfidf_vect_ngram = TfidfVectorizer(lowercase = True, analyzer='word', token_pattern=r
                                            ngram_range=(1,3), max_features=500)
         tfidf_vect_ngram.fit(df['text'])
         xtrain_tfidf_ngram =  tfidf_vect_ngram.transform(train_x)
         xvalid_tfidf_ngram =  tfidf_vect_ngram.transform(valid_x)
```

```
In [36]: # Linear Classifier on Ngram Level TF IDF Vectors
         # C (penalty) : 1 (Default)
         # Solver - Liblinear (Default)
         # Multiclass - OVR (one versus rest)
         # Default for max_iter is 100 which means that
         # the solver either coverges within 100 iteration or stops after 100 iterations
         results_ngram = train_model_weighted(LogisticRegression(), xtrain_tfidf_ngram, train_
         print ("LR, N-Gram Vectors TF-IDF train accuracy: ", results_ngram[1][0])
         print("")
         print ("LR, N-Gram Vectors TF-IDF train precision: ", results_ngram[1][1])
         print("")
         print ("LR, N-Gram Vectors TF-IDF train recall: ", results_ngram[1][2])
         print("")
         print ("LR, N-Gram Vectors TF-IDF train f1_score: ", results_ngram[1][3])
         print("***********************************************************")
         print ("LR, N-Gram Vectors TF-IDF test accuracy: ", results_ngram[0][0])
         print("")
         print ("LR, N-Gram Vectors TF-IDF test precision: ", results_ngram[0][1])
```

```
        print("")
        print ("LR, N-Gram Vectors TF-IDF test recall: ", results_ngram[0][2])
        print("")
        print ("LR, N-Gram Vectors TF-IDF test f1_score: ", results_ngram[0][3])
```

```
LR, N-Gram Vectors TF-IDF train accuracy:  0.60528

LR, N-Gram Vectors TF-IDF train precision:  0.5717103914739755

LR, N-Gram Vectors TF-IDF train recall:  0.60528

LR, N-Gram Vectors TF-IDF train f1_score:  0.570436947835613
**********************************************************
LR, N-Gram Vectors TF-IDF test accuracy:  0.602632

LR, N-Gram Vectors TF-IDF test precision:  0.5676778741628343

LR, N-Gram Vectors TF-IDF test recall:  0.602632

LR, N-Gram Vectors TF-IDF test f1_score:  0.5677683795162342
```

0.5 Model 3 : Bag of word representation - Ngram level 1-3 grams : Change Solver, multi_class

In [22]:
```
# Linear Classifier on ngram level TF IDF Vectors
# C (penalty) : 1 (Default)
# Solver (Optimization algorithm) : Saga
# multi_class : multinomial
# maximum iterations 10000
# Weighted accuracy, precision, recall, f-score
new_results_ngram = train_model_weighted(LogisticRegression(solver = "saga", multi_cla
                                    xtrain_tfidf_ngram, train_y, xvalid_tfidf_ngram)
print("PERFORMANCE METRICS WITH AVERAGE = 'WEIGHTED'")
print ("LR, N-Gram Vectors TF-IDF train accuracy: ", new_results_ngram[1][0])
print("")
print ("LR, N-Gram Vectors TF-IDF train precision: ", new_results_ngram[1][1])
print("")
print ("LR, N-Gram Vectors TF-IDF train recall: ", new_results_ngram[1][2])
print("")
print ("LR, N-Gram Vectors TF-IDF train f1_score: ", new_results_ngram[1][3])
print("**********************************************************")
print ("LR, N-Gram Vectors TF-IDF test accuracy: ", new_results_ngram[0][0])
print("")
print ("LR, N-Gram Vectors TF-IDF test precision: ", new_results_ngram[0][1])
print("")
print ("LR, N-Gram Vectors TF-IDF test recall: ", new_results_ngram[0][2])
print("")
```

```
        print ("LR, N-Gram Vectors TF-IDF test f1_score: ", new_results_ngram[0][3])
```

```
PERFORMANCE METRICS WITH AVERAGE = 'WEIGHTED'
LR, N-Gram Vectors TF-IDF train accuracy:  0.609176

LR, N-Gram Vectors TF-IDF train precision:  0.5811432125520433

LR, N-Gram Vectors TF-IDF train recall:  0.609176

LR, N-Gram Vectors TF-IDF train f1_score:  0.5867178933712871
************************************************************
LR, N-Gram Vectors TF-IDF test accuracy:  0.60808

LR, N-Gram Vectors TF-IDF test precision:  0.5790160769232071

LR, N-Gram Vectors TF-IDF test recall:  0.60808

LR, N-Gram Vectors TF-IDF test f1_score:  0.5848200240762588
```

0.6 Model 4 : Bag of word representation - Ngram level 1-3 grams, Solver, multi_class same as in model 3 : Change C (penalty) to 10

```
In [23]: # Linear Classifier on ngram level TF IDF Vectors
         # C (penalty) : 10
         # Solver (Optimization algorithm) : Saga
         # multi_class : multinomial
         # maximum iterations 10000
         results_ngram_penalty = train_model_weighted(LogisticRegression(C = 10, solver = "saga
                                     xtrain_tfidf_ngram, train_y, xvalid_tfidf_ngram)
         print("PERFORMANCE METRICS WITH AVERAGE = 'WEIGHTED'")
         print ("LR, N-Gram Vectors TF-IDF train accuracy: ", results_ngram_penalty[1][0])
         print("")
         print ("LR, N-Gram Vectors TF-IDF train precision: ", results_ngram_penalty[1][1])
         print("")
         print ("LR, N-Gram Vectors TF-IDF train recall: ", results_ngram_penalty[1][2])
         print("")
         print ("LR, N-Gram Vectors TF-IDF train f1_score: ", results_ngram_penalty[1][3])
         print("************************************************************")
         print ("LR, N-Gram Vectors TF-IDF test accuracy: ", results_ngram_penalty[0][0])
         print("")
         print ("LR, N-Gram Vectors TF-IDF test precision: ", results_ngram_penalty[0][1])
         print("")
         print ("LR, N-Gram Vectors TF-IDF test recall: ", results_ngram_penalty[0][2])
         print("")
         print ("LR, N-Gram Vectors TF-IDF test f1_score: ", results_ngram_penalty[0][3])
```

```
PERFORMANCE METRICS WITH AVERAGE = 'WEIGHTED'
LR, N-Gram Vectors TF-IDF train accuracy:  0.6090426666666666
```

```
LR, N-Gram Vectors TF-IDF train precision:  0.5813957819403074

LR, N-Gram Vectors TF-IDF train recall:  0.6090426666666666

LR, N-Gram Vectors TF-IDF train f1_score:  0.5874130996329474
************************************************************
LR, N-Gram Vectors TF-IDF test accuracy:  0.608536

LR, N-Gram Vectors TF-IDF test precision:  0.5799956513928765

LR, N-Gram Vectors TF-IDF test recall:  0.608536

LR, N-Gram Vectors TF-IDF test f1_score:  0.5860675509942822
```

0.7 Model 5 : Bag of word representation - Ngram level 1-3 grams, Solver, multi_class same as in model 3 : Change C (penalty) to 0.1

In [24]:
```
# Linear Classifier on ngram level TF IDF Vectors
# C (penalty) : 0.1
# Solver (Optimization algorithm) : Saga
# multi_class : multinomial
# maximum iterations 10000
results_ngram_penalty = train_model_weighted(LogisticRegression(C = 0.1, solver = "sag
                                    xtrain_tfidf_ngram, train_y, xvalid_tfidf_ngram)
print("PERFORMANCE METRICS WITH AVERAGE = 'WEIGHTED'")
print ("LR, N-Gram Vectors TF-IDF train accuracy: ", results_ngram_penalty[1][0])
print("")
print ("LR, N-Gram Vectors TF-IDF train precision: ", results_ngram_penalty[1][1])
print("")
print ("LR, N-Gram Vectors TF-IDF train recall: ", results_ngram_penalty[1][2])
print("")
print ("LR, N-Gram Vectors TF-IDF train f1_score: ", results_ngram_penalty[1][3])
print("************************************************************")
print ("LR, N-Gram Vectors TF-IDF test accuracy: ", results_ngram_penalty[0][0])
print("")
print ("LR, N-Gram Vectors TF-IDF test precision: ", results_ngram_penalty[0][1])
print("")
print ("LR, N-Gram Vectors TF-IDF test recall: ", results_ngram_penalty[0][2])
print("")
print ("LR, N-Gram Vectors TF-IDF test f1_score: ", results_ngram_penalty[0][3])
```

```
PERFORMANCE METRICS WITH AVERAGE = 'WEIGHTED'
LR, N-Gram Vectors TF-IDF train accuracy:  0.6054053333333334

LR, N-Gram Vectors TF-IDF train precision:  0.5745269292835176
```

```
LR, N-Gram Vectors TF-IDF train recall:  0.6054053333333334

LR, N-Gram Vectors TF-IDF train f1_score:  0.5778688241565018
************************************************************
LR, N-Gram Vectors TF-IDF test accuracy:  0.603632

LR, N-Gram Vectors TF-IDF test precision:  0.5713922168741851

LR, N-Gram Vectors TF-IDF test recall:  0.603632

LR, N-Gram Vectors TF-IDF test f1_score:  0.5750064993168574
```

In [ ]: