Assignment 3: Text Classification Report

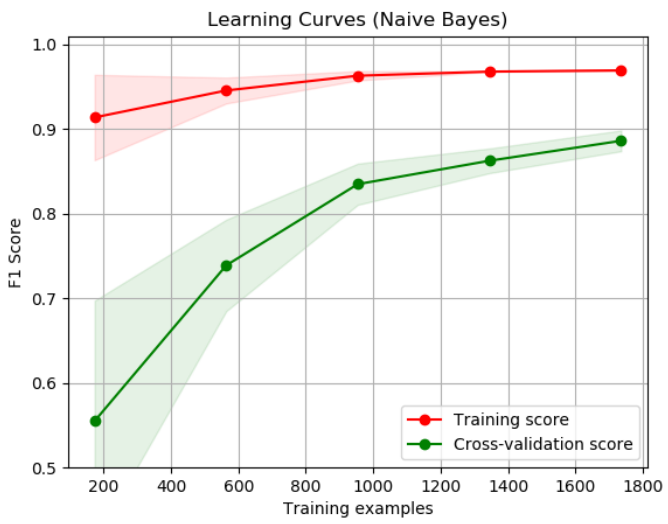
Sharad Sridhar

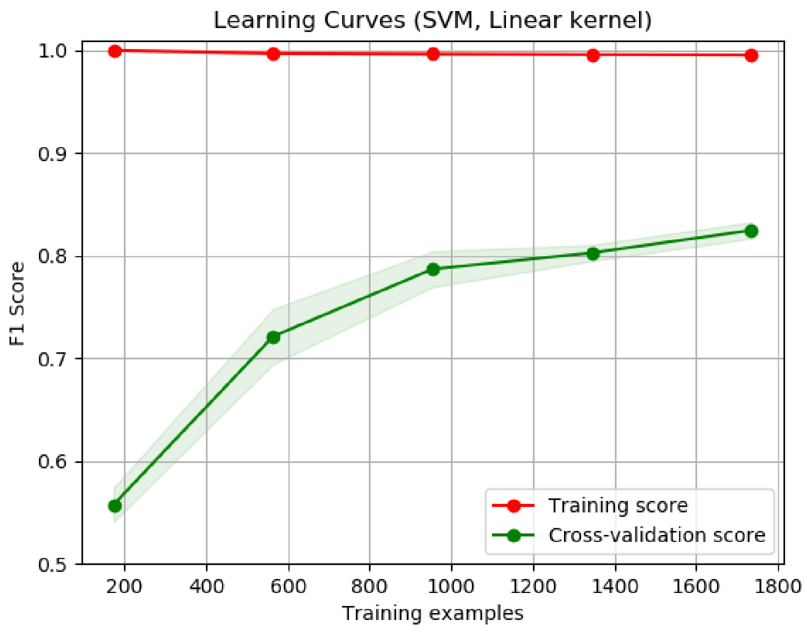
111492675

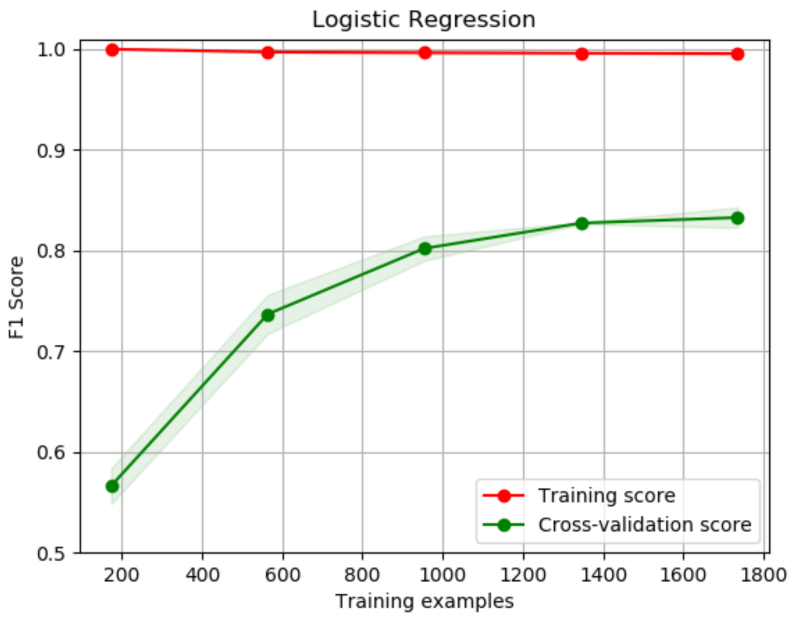
Output for the Macro-Average of precision / recall and F1 values

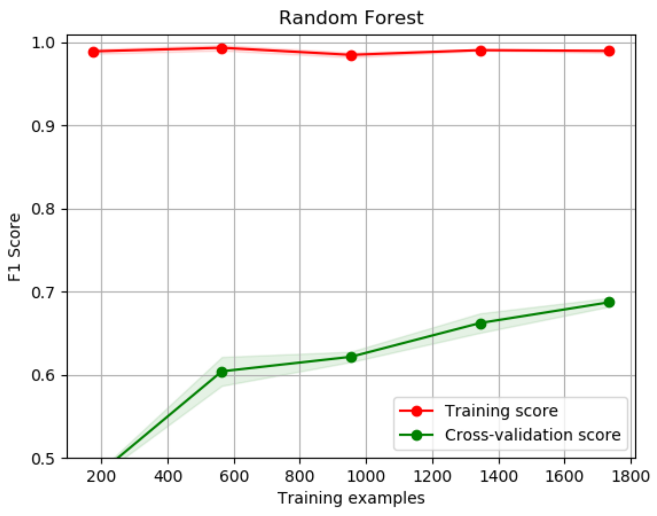
```
NB,UB,0.861,0.833,0.840

NB,BB,0.835,0.756,0.768

LR,UB,0.785,0.781,0.781

LR,BB,0.768,0.744,0.749

SVM,UB,0.777,0.769,0.771

SVM,BB,0.744,0.710,0.714

RF,UB,0.709,0.667,0.661

RF,BB,0.666,0.602,0.595
```

Learning Curves




Learning Curves (SVM, Linear kernel)
F1 Score
Training examples
Training score
Cross-validation score
1.0
0.9
0.8
0.7
0.6
0.5
200
400
600
800
1000
1200
1400
1600
1800


Logistic Regression
F1 Score
Training examples
Training score
Cross-validation score
1.0
0.9
0.8
0.7
0.6
0.5
200
400
600
800
1000
1200
1400
1600
1800



OBSERVATIONS:

- The most obvious thing noticed here is that the score increases as the size of the data increases. This is simply because the classification model gets to train and learn on a larger data-set.
- The best result was obtained from the Multinomial Naïve Bayes classifier (unigram)
- The data being worked upon is relatively simple. There are no heavy dependencies. We are merely classifying words. Even more importantly, since, unlike the others, NB works quite well on relatively smaller training datasets, it becomes more suited to our task as we are operating only on a small portion of the data-set
- It is also a very straightforward classifier, which is not particularly sensitive to irrelevant features.
- Although the other algorithms are more ‘powerful’, they are probably not ‘trained’ enough to produce better results. They need large amounts of training data.

MBC_Exploration results with Multinomial Naïve Bayes:

```
NB+UB,CountVectorizer,0.866,0.864,0.864
NB+UB,TFIDFVectorizer,0.894,0.877,0.883
NB+UB,CountVectorizer+StopWord,0.861,0.858,0.859
NB+UB,TFIDFVectorizer+StopWord,0.893,0.879,0.884
NB+UB,CountVectorizer+StopWord+Stemming,0.868,0.869,0.868
NB+UB,TFIDFVectorizer+StopWord+Stemming,0.881,0.869,0.873
NB+UB,CountVectorizer+Stemming,0.864,0.865,0.864
NB+UB,TFIDFVectorizer+Stemming,0.876,0.865,0.868
```

MBC_Final result with Multinomial Naïve Bayes:

0.884097021925

Explanation:

- For this, I switched the CountVectorizer to TFIDFVectorizer.
- CountVectorizer is just occurrence count, which leads to the situation that longer documents will have higher average count values than shorter documents, even though they might talk about the same topics.
- To avoid these potential discrepancies, it suffices to divide the number of occurrences of each word in a document by the total number of words in the document: these new features are called tf for Term Frequencies.
- Another refinement on top of tf is to downscale weights for words that occur in many documents in the corpus and are therefore less informative than those that occur only in a smaller portion of the corpus. Thus, we get our TFIDVectorizer.
- I also reduced the alpha value for the classifier to lessen smoothing which led to higher scores.
- As can be seen, stemming somewhat reduces the score. This could be due to improper stemming of a word. Sometime at various places, the same word could be different in its meaning.
- Removing stop words is beneficial in trimming unnecessary data from the set. They have no sematic significance and essentially clutter up the info.

References:

https://stackoverflow.com/questions/5791581/how-can-i-combine-multiple-lines-of-text-into-one-line-in-python-with-a-delimite

https://stackoverflow.com/questions/6181763/converting-a-string-to-a-list-of-words

https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a

http://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html

http://scikit-learn.org/stable/modules/learning_curve.html

Some other portions were used directly from the SCIKIT_LEARN website as given. Some general discussions about reading info from the files and general approach were done with fellow classmates.