Artificial Intelligence HW 3
Kondipati Venkata Neeharika
110346722

Q1 Basic Comparison with Baselines

a. Macro Average of precision, recall and F1 Scores for 4 classifiers

Unigrams	Precision	Recall	F1-Score
Naive Bayes	0.863937	0.736684	0.705525
Logistic Regression	0.900183	0.851460	0.854937
SVM	0.895258	0.889221	0.891456
Random Forests	0.785891	0.724105	0.712265
Bigrams	Precision	Recall	F1-Score
Naive Bayes	0.868760	0.745747	0.737506
Logistic Regression	0.874031	0.797888	0.797701
SVM	0.877806	0.863602	0.867679
Random Forests	0.706313	0.650236	0.643661

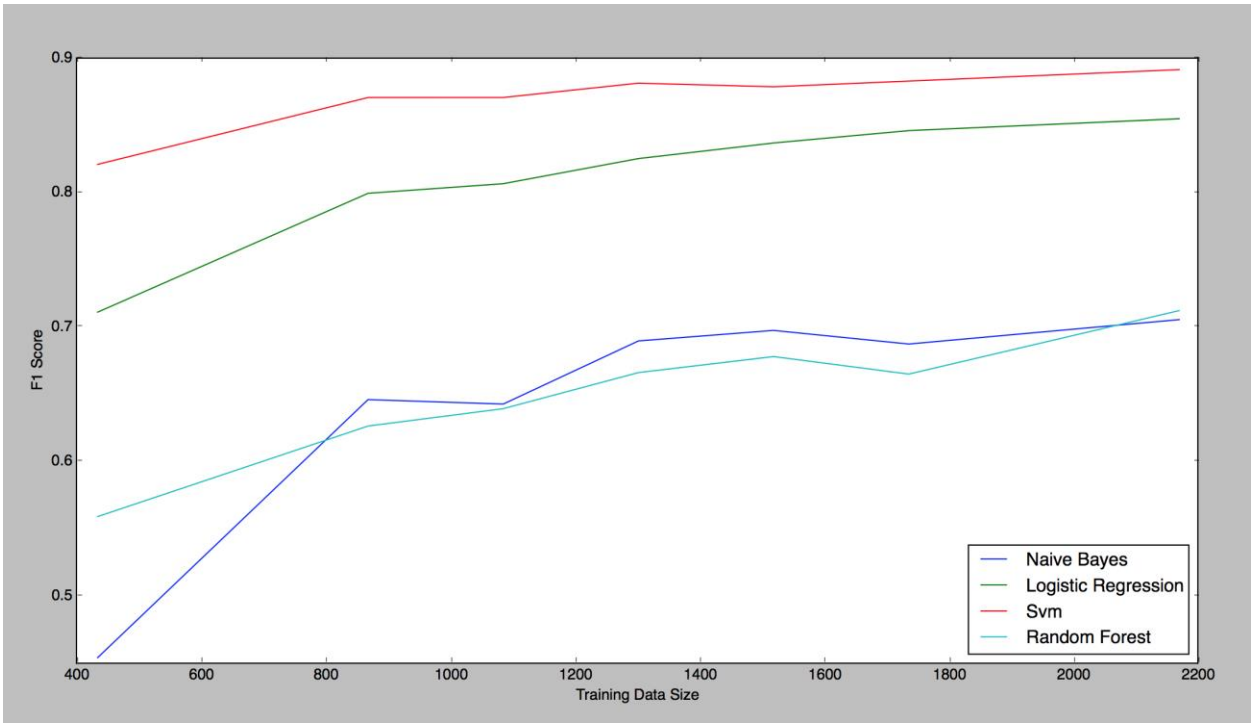
For the given text classification problem, SVM with a linear kernel out performs other classifiers .i.e

svm > logistic Regression > Random Forests > naive Bayes

We observe that uni grams give better results than bi grams , which is because of data sparsity i.e as the n-gram length increases, the amount of times we will see any given n-gram will decrease.

Also because we have a very relatively large amount of vocabulary in the documents but each of these types has a very low frequency, we get better results with uni-gram model.

b. Learning curve (Training Data Size vs F1 Score of the Classifier)



c. Findings and arguments on the results

Train and test data is transformed into count vectors of unigrams and bigrams using count vectorizer (considers the frequency of the words in all the documents) and then Tf-idf is applied on the count vectors.

Among the four classifiers , SVM with linear kernel seems to work better than the others. Almost always text classification is always linearly separable hence the best result for svm linear kernel.

In comparison to other classifiers,
Major dis advantage of **Naive Bayes** is that it can't learn interactions between features because of independence assumption (which does not hold in case of text

classification). Also without tf-idf transformer Naive Bayes gives good results compared to other classifiers which is because :
Data set specific reason : Class overlapping is small in this dataset(linearly separable)
Tf-idf : increases the importance of few words disturbing the bag of words probability of the naive bayes classifier thus giving poor results.

Linear SVMs and Logistic Regression generally perform comparably but in this case we get better results with SVM because the data being linearly separable,
maximizing the margin gives better results than maximizing the posterior class probability which is the case in logistic regression.

Random forests work better with data whose features do not interact linearly and designed to handle high dimensional spaces and large training data which is not the case we are handling, therefore linear svm out performs random forests.

Q2. My Best Configuration

a. Exploration results of different Configurations of SVM

Configurations	Precision	Recall	F1 Score
Stop Words + stemming	0.90450058072009287	0.8964299491686426	0.8995593232215825
Stop Words + l1 Regularization	0.86566139700913469	0.85982666760933091	0.85578552198347524
Stop Words + l2 Regularization	0.90797155062507617	0.85510480795028532	0.86153349426698833
Stemming + l1 Regularization	0.87382933809154695	0.86449732428375659	0.86729225364823115
Stemming + l2 Regularization	0.90241768719988524	0.90261346414612742	0.90245442789186003
Stop Words + stemming + l1 Regularization	0.87432782017770938	0.87445583799227022	0.87318761805145928
Stop Words + stemming + l2 Regularization	0.8975936645676772	0.87764542409014767	0.88309330117400031
StopWords + Rbf Kernel	0.069156293222683268	0.25	0.10834236186348864
StopWords + Polynomial Kernel	0.069156293222683268	0.25	0.10834236186348864
Stop Words + l2 Regularization + Regularization const (= 0.0001)	0.887653	0.890082	0.880939
Stop Words + l2 Regularization + Regularization const (= 0.0005)	0.914922	0.879952	0.888454
*Stop Words + Stemming + l2 Regularization + Regularization const (= 0.2e-3)	0.913131	0.919426	0.914562

b. Best configuration training model :

Way to run the python code hw3.py :

Flag = 0 for my best configuration (where training and test performed at one place)

```
python hw3.py <Training_Folder> <Test_Folder> 0
Flag = 1 for all the classifiers analysis and learning curve
python hw3.py <Training_Folder> <Test_Folder> 1
```

Way to run the python code train.py :

```
Trains the best configuration model and exports into
model_.pickle file
python train.py <Training Folder>
```

Way to run the python code test.py :

```
Predicts the test documents on the best configuration model by
reading from the model_.pickle file
```

python test.py <Test Folder>

c. **Explanation :**

For the best configuration, the following design choices have been made :

Removal of Stop Words
Stemming of Words
Linear SVM (SDGC with linear classifier i.e loss = 'hinge')
L2 regularization
Alpha = 0.2e-2

Stop words and stemming helped in enhancing the feature vectors, in our case removal of stop words and stemming helped reduce the number of features by 1869, where actual corpus had 34267 words.
Stemming helped in matching related word to the same stem thus combining the occurrences of similar words.
Linear SVM outperformed other classifiers because the data given is linearly separable.
Usually Regularization is a factor use to prevent over fitting. L2 is usually sum of squares of weights which is added to the weight function. L2 is the standard regularizer for linear SVM models which seems to work better in our case.
Alpha is the regularization constant that multiplies the regularization term which is chosen to be 0.2e-2.

Citations :

-- Sklearn libraries – documentation
--http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/
--Stack overflow