# Assignment 3
# Report

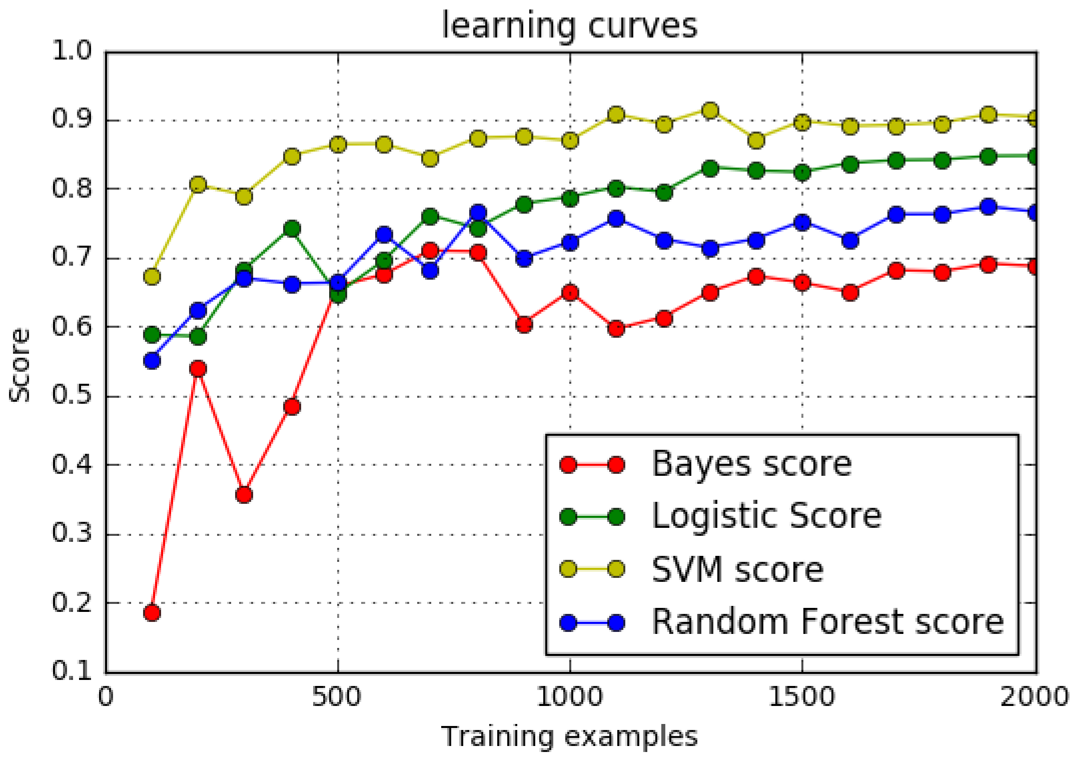
**Basic Comparison with Baselines(50 points) :**

1. (20 points)A table of macro-average of precision/recall and F1 values for all 8 runs (4 classifiers x two representations)

| **Classifier** | **Representation** | **Precision** | **Recall** | **F1** | **time** |
|---|---|---|---|---|---|
| Naive Bayes | Unigram | 0.862 | 0.725 | 0.691 | 0.843 |
| | Bigram | 0.867 | 0.769 | 0.753 | 2.013 |
| SVM | Unigram | 0.911 | 0.90 | **0.904** | 0.858 |
| | Bigram | 0.903 | 0.874 | **0.882** | 2.505 |
| Logistic Regression | Unigram | 0.896 | 0.845 | 0.849 | 1.078 |
| | Bigram | 0.876 | 0.797 | 0.796 | 3.13 |
| Random Forest | Unigram | 0.867 | 0.769 | 0.7538 | 4.11 |
| | Bigram | 0.809 | 0.719 | 0.713 | 15.78 |

Best performing classifier among all the classifiers is **SVM.**

**2.** (20 points) A learning curve result, where you show the performance of each classifier with just the unigram representation. The learning curve is a plot of the performance of the classifier (F1 on the y-axis) on the dev fold, when trained on different amounts of training data (size of training data on the x-axis).



**3.** (10 points) Describe your findings and make arguments to explain why this is the case. You can use any online source to understand the relative performance of algorithms for varying training data sizes. Make arguments in your own words. Provide a citation

There can be many ways to optimize a model and we cannot generalize that a single classification algorithm is best. Different classifiers should be considered and compared for a given dataset.

**Naive Bayes** is often used as a baseline model in text classification because it is fast and easy to implement. If we see the above results for almost all the samples  F1 score from Naive Bayes the least among all the models and also its  performance is very bad for small size training samples (if you can see the training sample of size 100 naive bayes has 0.19 as its F1 score) and increases as the sample size increases. It implies that one should train a model with large training dataset to get good results from Naive Bayes. Also it is relatively fast compared to other classification algorithms( takes 0.84 sec for unigrams and 2 sec for Bigrams).

**SVM** performs best among all the algorithms and takes less amount of time(approximately equal to Naive Bayes).They provide high accuracy with the varied sample sizes compared to other classification algorithms(From the figure it is clearly evident). Also Naive Bayes does not provide optimal results when samples are correlated and these cases are handled well by SVM. Thus we can use SVM irrespective of the size of the training samples.

**Logistic Regression** performs better than Naive Bayes as we need not worry about features being correlated and also it is simple compared to SVM and Random Forest. From the figure it is evident that logistic regression performs reasonably well. Thus we can use it in the scenarios where we need simple implementation and  can compromise some level of accuracy.

**Random Forest** when tuned with proper hyper parameters provide good results, most of the times better than SVM. But in the above example as we are not tuning any hyper parameters therefore there may be overfitting of the sample which might be resulting in relatively low score compared to SVM.
**Source:** https://www.quora.com/What-are-the-advantages-of-different-classification-algorithms

**My best configuration (50 points)** -- Pick the best performing classification algorithm from the above experiments and then use the design choices below to find the best possible result on the dev set. You will create a model that we can run on a hidden test data.

a. Exploration results (20 points) -- You should provide results from 8 configurations picking two choices from each of the options below

The best performing classification algorithm from the above experiments is SVM.

The following are different configurations of best  performing classification algorithm:
**CountVectorizer + TfidfTransformer** is applied for all the models

| **Model** | **Configuration** | **Precision** | **Recall** | **F1** |
|---|---|---|---|---|
| 1 | Stop Words + L2 Penalization (Best Configuration) | 0.923 | 0.9122 | 0.9125 |

| | | | | |
|---|---|---|---|---|
| 2 | Porter Stemmer + Stop Words + L2 Penalization | 0.9293 | 0.91604 | 0.908 |
| 3 | Porter Stemmer + Stop Words + L1 Penalization | 0.903 | 0.8894 | 0.8941 |
| 4 | Porter Stemmer only | 0.9122 | 0.9038 | 0.906 |
| 5 | Porter Stemmer + Stop Words | 0.915 | 0.900 | 0.905 |
| 6 | Porter Stemmer + Stop Words + L1 based Feature Selection + L1 Penalization | 0.9135 | 0.8951 | 0.9008 |
| 7 | Porter Stemmer + Stop Words + L1 based Feature Selection + L2 Penalization | 0.919 | 0.901 | 0.9073 |
| 8 | Porter Stemmer + Stop Words + Univariate Feature Selection + L2 Penalization | 0.920 | 0.903 | 0.909 |
| 9 | Porter Stemmer + Stop Words +Tree Based Feature Selection + L2 Penalization | 0.918 | 0.900 | 0.9066 |

b. Code (20 points)-- Export the best configuration training model and give us code that we can run to get performance of your model on a test set. Make sure to write a brief documentation on how to run your code in your report. The organization of the test data we use is the same as the test data you are given, you can design an interface accordingly.

**Documentation:**
My Best configuration is obtained by removing stop words with L2 penalization on SVM.
I have used SGDclassifier with hinge loss ( a linear SVM, as the number of features is more than 10000 most likely points are linearly separable ) as the classifier with above mentioned.

I have created two python files one to build a model and another to test the model:

Execute the following command to pre-learned model run on a test dataset:

```
python best_config_test.py <Testing_Samples_Location>
```

Execute the following command to get a learned model from a training dataset:

```
python best_config_train.py <Testing_Samples_Location>
```

(I have included other files training.pkl (Prelearned model), Analysis_config.py (Exploration for best params), Learning_curves.py(Code for Learning curves plot ), Unigram_bigram_models.py (baseline models and all work with command: code.py <Training_samples_location> <Testing_Samples_Location>)

c. Explanation (10 points) -- Explain your result based on your best understanding of the configuration options. Should be no longer than 2 paragraphs. You can use any online source to understand the options and what they mean. Provide a citation of the sources.

The Best configuration obtained among all the options is using **Stop Words + L2 Penalization** + **CountVectorizer + TfidfTransformer** combination with SVM. L2 penalized SVM can capture models that require many tiny features and as stop words are the most commonly used words and removing these words, helps us focus on the important words.TFID Transformer reduces the weightage of commonly used words thus giving preference to important words. Thus combining these design choices we can capture even the most granular features from important words which helps us build the model using necessary components and avoiding the redundant ones resulting the high performance.

In most of the cases including stemmer will increase performance but in this case it reduced. It may happen that all words like run are in one category and running in another category, thus there will be misclassification of the words. L1 penalization or L1 based feature selection reduces the number of features, this may remove certain prominent features (for example there may be a word hindu in talk.religion.misc with less frequency but it is very prominent in classification as we can clearly say that hindu belongs to talk.religion.misc category, thus removing such words might reduce the performance. Same is the case with univariate feature selection which works by selecting the best features based on univariate statistical tests and tree based feature selection discard irrelevant features.

**Sources:** http://scikit-learn.org/stable/modules/feature_selection.html
https://www.quora.com/What-are-the-advantages-of-different-classification-algorithms