MESSAGE SPAM DETECTION

CAPSTONE PROJECT

MACHINE LEARNING ENGINEER NANODEGREE

ABHISHEK VERMA
github.com/hell-sing

Table of Content

I. Definition

This section defines the problem statement as well as an overview of the message spam detection. The metrics used for evaluating the model and the feature set are also defined below.

1.1. Project Overview

The purpose of this paper is to explore the results of applying machine learning techniques to Message spam detection. Short Message Service spam (sometimes called cell phone spam) is any junk message delivered to a mobile phone as text messaging through the SMS. The dataset for this project originates from the UCI Machine Learning Repository [1]. More detail about dataset can be found on UCI dataset official website.

- This dataset has been collected from free or free for research sources at the Internet.
- The collection is composed of just one text file, where each line has the correct class followed by the raw message.

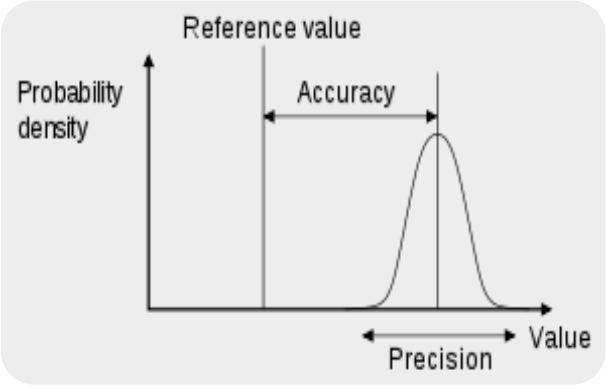
1.2. Problem statement

In the paper, we would try to analysis different methods to identify spam messages. We will use the different approach, based on word count and term-frequency inverse document-frequency (tf-idf) transform to classify the messages. Following steps are required in order to achieve the objective:

1. Download and pre-process the SMS Spam Collection v.1 dataset.
2. Test and find best approach (word count or tf-idf vectorizer) to classify the messages.
3. Selection of approach and splitting the dataset into training and testing data.
4. Initialize various classifier and train it.
5. Evaluate the classifiers and finding best the model for a dataset.

1.3. Metrics

Accuracy is the first metric to be checked when the algorithms are evaluated, is the sum of true positives and the true negative outputs divided by the data size.



As shown in the image on the right accuracy means how closer you are to the true value, whereas precision means that your data points are not widely spread.

The **Scikit-learn** library provides a convenience report when working on classification problems to give you a quick idea of the accuracy of a model using a number of matrices, one of them is **F1_score** which work with all the model [2].

Accuracy is defined as the number of true positives (TP) plus the number of true negative (TN) over the total number of sample (N).

$$accuracy(total) = \frac{TP + TN}{N}$$

Precision (P) is defined as the number of true positives (TP) over the number of true positives plus the number of false positives (FP) whereas **Recall (R)** is defined as the number of true positives (TP) over the number of true positives plus the number of false negatives (FN).

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN}$$

These quantities are also related to the **F1 score**, which is defined as the harmonic mean of precision and recall. I found F1_score as appropriate to use as report metric in order to have a good idea of how the algorithm is behaving.

$$F1 = 2\frac{P \times R}{P + R}$$

II. Analysis

Below describes how the data was gathered, which features were selected, and which algorithms were explored. Finally, I outline the benchmark used to evaluate the performance of the trading strategy.

2.1. Data exploration

The dataset used for this project is SMS Spam Collection v.1 dataset originates from the UCI Machine Learning Repository. This dataset has been collected from free or free for research sources at the Internet. The collection is composed of just one text file, where each line has the correct class followed by the raw message.

	Class	Text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
Number of ham messages in data set: 4825

Number of spam messages in data set: 747
```

This dataset is tab-separated values (TSV) file. There are total 5572 entries in the dataset and has two column “Class” and “Text” where each row represent different message and Class contain two unique categories ham and spam. Dataset does not require any kind of cleaning, wrangling and there is no null value in any column.

2.2 Exploratory Visualization

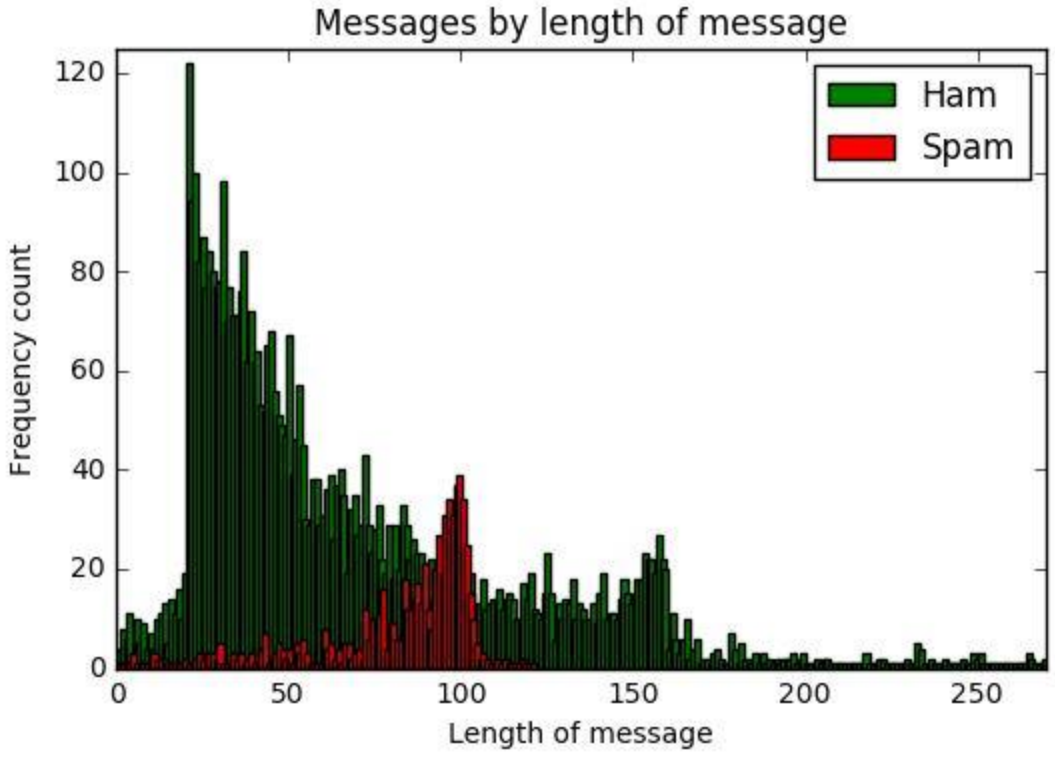
The 2 categories ham and spam, it's possible to take advantage of the text since it is different in each of the categories. There are a lot of words common words in the same class of messages. Sample data from both classes has been provided below:

	Class	Text
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
5	spam	FreeMsg Hey there darling it's been 3 week's n...
8	spam	WINNER!! As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...
11	spam	SIX chances to win CASH! From 100 to 20,000 po...

	Class	Text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
6	ham	Even my brother is not like to speak with me. ...

Since each category of messages has some similar type of keywords i.e. in spam message words like Free, Winner, win, won, award, congrats, congratulation, selected, urgent, and Cash it is easier to classify them.

Another way of looking messages is word count, counting words in each message and plotting graph with the increasing number of word length. Spam messages are plotted in red and ham messages are plotted in green. But since all the spam messages are overlapping on ham messages it would be very difficult to identify spam/ham messages solely on the basis of length. Thus classify on the basis of word count would not be useful.



2.3 Algorithms and Techniques

Text classification problems can be both supervised and unsupervised. As we explained earlier, our problem context made us select an approach based on supervised learning, where we wanted to classify our text and check whether it is spam or ham. The classification problems offer a wide variety of machine learning algorithms to solve a problem, it's impossible to know which technique and which algorithm is going to work the best way on the dataset beforehand. That means we need to using trial and error.

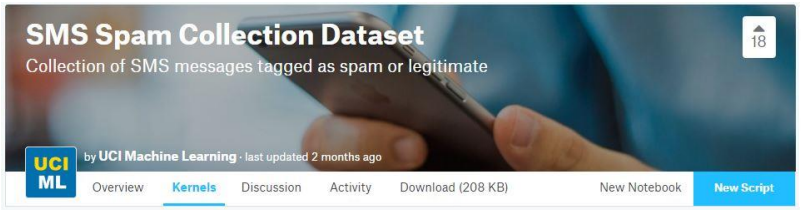
Techniques which we are going to us are based on **word count** and term-frequency inverse document-frequency **(tf-idf) transform**.

Some of the most important machine learning algorithms are included in this project are **Naive Bayes, Decision Tree, AdaBoost, K-Nearest Neighbours and Random Forest**.

- **Naive Bayes (Multi-NB)** one of the simplest classification algorithms that exist. Naive Bayes doesn't require a ton of data and is known to be very fast.
- **Decision Trees (DTs)** are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
- **AdaBoost (AdaBoost)** classifier is a meta-estimator (in our case it is Decision Tree) that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.
- **Random Forest (RF)** is easy to interpret and is nonparametric, meaning we don't have to worry about tuning a bunch of parameters such as when using a Support Vector Machine. They are often praised because they work "out of the box".
- **KNearest Neighbor (KNN)** a simple and often effective classification or regression algorithm. K-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification.

2.4 Benchmark

Testing the accuracy of the previous kagglers: The result shows that Naïve Bayes work better on the dataset with an accuracy_score of 0.90 our benchmark for the project will be Kaggle [3].




```
In [5]: nb_classifier = GaussianNB()
        nb_classifier.fit(x_train_tf, y_train)
        y_predict = nb_classifier.predict(x_test_tf)

In [6]: print(accuracy_score(y_test, y_predict))
        0.904306220096
```

III. Methodology

3.1 Data Pre-processing

As specified before dataset does not require any sort of cleaning. Column “Class” contain categorical value ham and spam since classifier require numeric value ham is replaced with 1 and spam is replaced with 0. Similarly two new columns are introduced "Count". Count contain a number of words in a given message.

	Class	Text	Count
0	0	Go until jurong point, crazy.. Available only ...	111
1	0	Ok lar... Joking wif u oni...	29
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	0	U dun say so early hor... U c already then say...	49
4	0	Nah I don't think he goes to usf, he lives aro...	61

3.2 Implementation

3.2.1 Data processing

To train our classifier we need to use **term frequency–inverse document frequency (tf–idf)**, it is a numerical statistic that is intended to reflect how important a word is to a document in a corpus. It is used as a weighting factor in information retrieval, text mining, and user modelling.

$$tf.idf(t,d) = tf(t,d) \times idf(t)$$

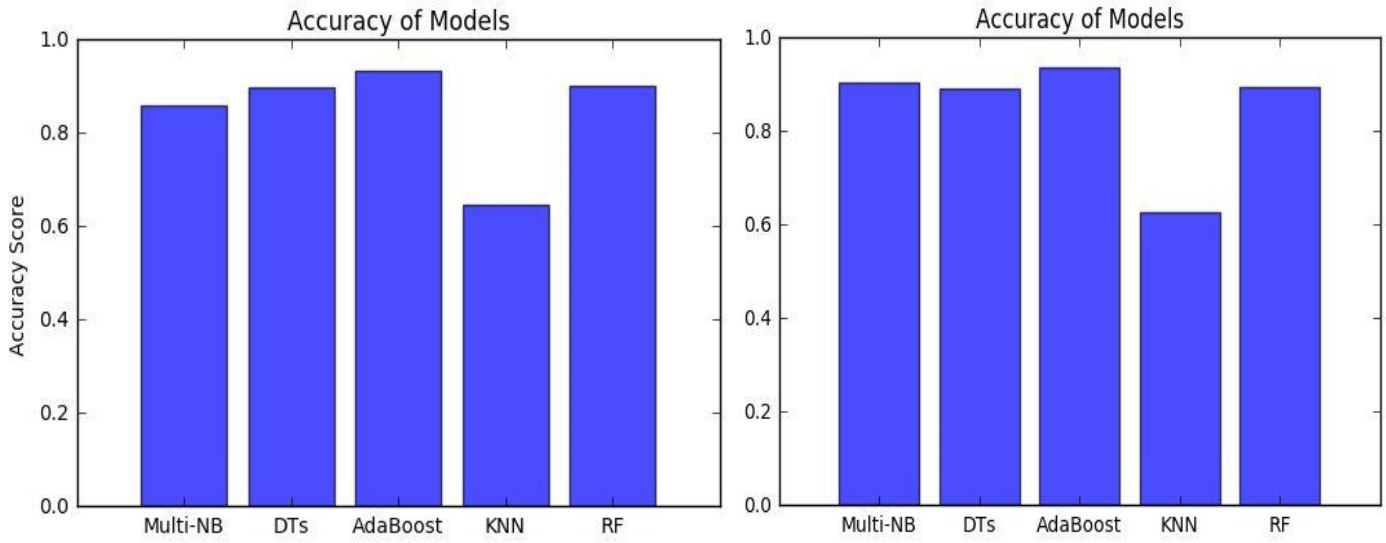
The tf-idf value increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

3.2.2 Classifier and Training

Before we start classifying, we first split the data into a training and testing set. Features are stored in X and target is stored in y. Data is divided into training and testing with a ratio of 4:1 which is 80% for training and 20% for testing. The classifiers are initialised for Naïve Bayes, Decision Tree, AdaBoost, Random Forest, and KNearest Neighbor and the training for the model starts, then the testing outputs the final predictions and compares them with the test labels to measure the accuracy using the accuracy_score. Training model becomes very simple without any due to TfidfVectorizer from **Scikit-learn** library, as it converts all the text data into numeric data and simplifies the process of training and testing for text data.

3.2.3 Visualization

The accuracy of the models after training them with model tuning and without model tuning. Bar graph has been plotted to show accuracy of different models. The bar graph on left shows the accuracy of models without tuning and model on right shows the accuracy of models with model tuning.



3.2.4 Results

Clearly, AdaBoost is working best when compared to all the other model, after that Random Forest, Decision Tree, then Naïve Bayes and KNearest Neighbor. Since all the models are working very well even without any type of model tuning or refinement there may be chances that models are over-fitting data thus archiving high accuracy.

3.3 Refinement

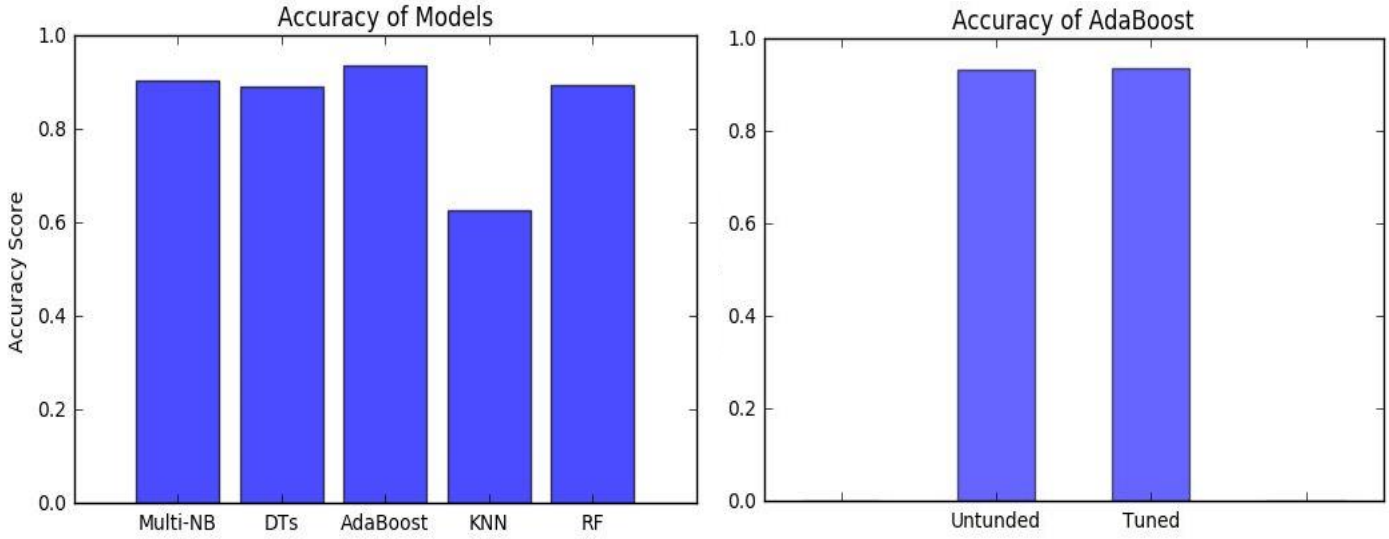
As mentioned earlier the tf-idf value increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. Words like I, me, my, it, the, he, she does not provide much information about ham/spam but are very high in frequency thus decreases the accuracy.

To remove these type of word from feature data, natural language toolkit (nltk) can be helpful, the nltk function for stopwords can be used to remove general words from feature data and provide more meaningful data for the classifier to train on and thus increases the an accuracy of model [4].

IV. Results

4.1 Model Evaluation and Validation

The accuracy of the models after training them with stopwords and model tuning. The bar graph has been plotted to show accuracy score of different models. Clearly the winner after refinement is **AdaBoost** with an accuracy of **0.9370** which better than untuned model 0.9347 and runner-up model is Naïve Bayes which have worked very well after refinement compared to the result before refinement. The bar graph on left shows accuracy after model tuning and bar graph on right show accuracy of AdaBoost with a tuned and unturned model. **No explicit tuning required** for AdaBoost as all the model has been tuned using **stop_word** before test_train_split.



4.2 Justification

The result of the models was very satisfactory, comparing to the benchmark. All the final models worked better than benchmark even with half the dataset, it is possible to say that the result is trustworthy and the project is finished under satisfaction the precision of 93%. The only thing negative is the time it takes for training, maybe 10 to 15 seconds but the output has correct labels and that is the most important.

V. Conclusion

5.1 Reflection

In this project, we tried to analysis different methods to identify spam messages. We used the different approach, based on word count and term-frequency inverse document-frequency (tf-idf) transform to classify the messages. Since test data have very high meaning for human and very difficult for the machine to understand, the biggest challenge was to analysis the test data and convert it into some meaningful numeric data without disturbing the relation between categories. Best method to convert the test data into meaningful numeric data is tf-idf vectorizer. This was the most interesting part of the whole project to convert huge amount of text data into numeric data. Best result was generated using tf-idf vectorizer with AdaBoost and Naïve Bayes classifier, both achieved accuracy approx. 93%, which is a satisfactory result.

5.2 Improvement

Finding an appropriate machine learning model is not the end of the work, it's possible to save and load the model using Scikit - learn's Pickle, a standard way to serializing objects in python to a file. A Neural network can achieve some great results [5]. This consideration must be taken in a count for improvement of the model.

VI. References

[1] http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/
[2] http://scikit-learn.org/stable/modules/classes.html
[3] https://www.kaggle.com/uciml/sms-spam-collection-dataset
[4] http://www.nltk.org/index.html