Problem Statement

SMS Spam Detection using Machine Learning Approach

Short Message Service (SMS) has grown rapidly and the reduction in the cost of messaging services has resulted in growth in unsolicited commercial advertisements (spams) being sent to mobile phones. Lack of real databases for SMS spams, short length of messages and limited features, and their informal language are the factors that may cause the established email filtering algorithms to underperform in their classification. A database of real SMS Spams from UCI Machine Learning repository is used, and after preprocessing and feature extraction, different machine learning techniques are applied to the database namely,

1) Multinomial Naïve Bayes
2) Support Vector Machine
3) K Nearest Neighbors
4) Random Forest
5) AdaBoost

and the results are compared.

Dataset Description

We use a database of 5574 text messages from UCI Machine Learning repository gathered in 2012. It contains a collection of 425 SMS spam messages manually extracted from the Grumbletext Web site (a UK forum in which cell phone users make public claims about SMS spam), a subset of 3,375 SMS randomly chosen non-spam (ham) messages of the NUS SMS Corpus (NSC), a list of 450 SMS non-spam messages collected from Caroline Tag's PhD Thesis, and the SMS Spam Corpus v.0.1 Big (1,002 SMS non-spam and 322 spam messages publicly available). The dataset is a large text file, in which each line starts with the label of the message, followed by the text message string. After preprocessing of the data and extraction of features, machine learning techniques such as naive Bayes, SVM, and other methods are applied to the samples, and their performances are compared.

Data Preprocessing, Feature Vector and Models Used:

1) The Dataset had two columns: 1st containing the class of data ham or spam and 2nd containing a string which is the text message.
2) All English Stopwords were imported from NLTK and were removed if found in the sentences.
3) We used a basic Count vectorizer from Sklearn library in order to tokenize and vectorize the string of text. The Count Vectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

   It can be used as follows:

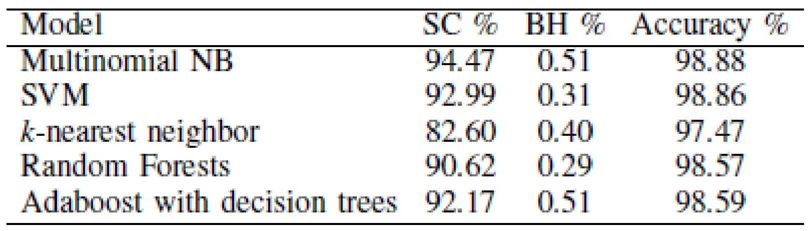
   a) Create an instance of the CountVectorizer class.
   b) Call the fit() function in order to learn a vocabulary from document.
   c) Call the transform() function on the document as needed to encode each as a vector.

   An encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document.

4) We got our numeric or real feature vector from string of text messages.
5) We next perform the test train split in the ratio of 70:30, we select the samples randomly.
6) We feed the X_train, X_test, y_train, y_test to different ML models namely, Multinomial naïve bayes, Support vector machines, K nearest neighbours, Random forest and AdaBoost.

Comparison of Outputs

Results obtained in research paper:



Model	SC %	BH %	Accuracy %
Multinomial NB	94.47	0.51	98.88
SVM	92.99	0.31	98.86
k-nearest neighbor	82.60	0.40	97.47
Random Forests	90.62	0.29	98.57
Adaboost with decision trees	92.17	0.51	98.59

Results obtained by our implemented models:

```
Unique values in the Class set:  ['ham' 'spam']

Number of ham messages in data set: 4825

Ham Count value 272

Number of spam messages in data set: 747

Spam Count value: 122


Training set has 3900 samples.

Testing set has 1672 samples.


Multi-NB

Accuracy in %:

97.84688995215312

F1 Score

0.9178082191780822


SVM

Accuracy in %:

98.14593301435407
```

F1 Score

0.9249394673123487

KNN

Accuracy in %:

95.15550239234449

F1 Score

0.7743732590529249

RF

Accuracy in %:

97.00956937799043

F1 Score

0.8724489795918368

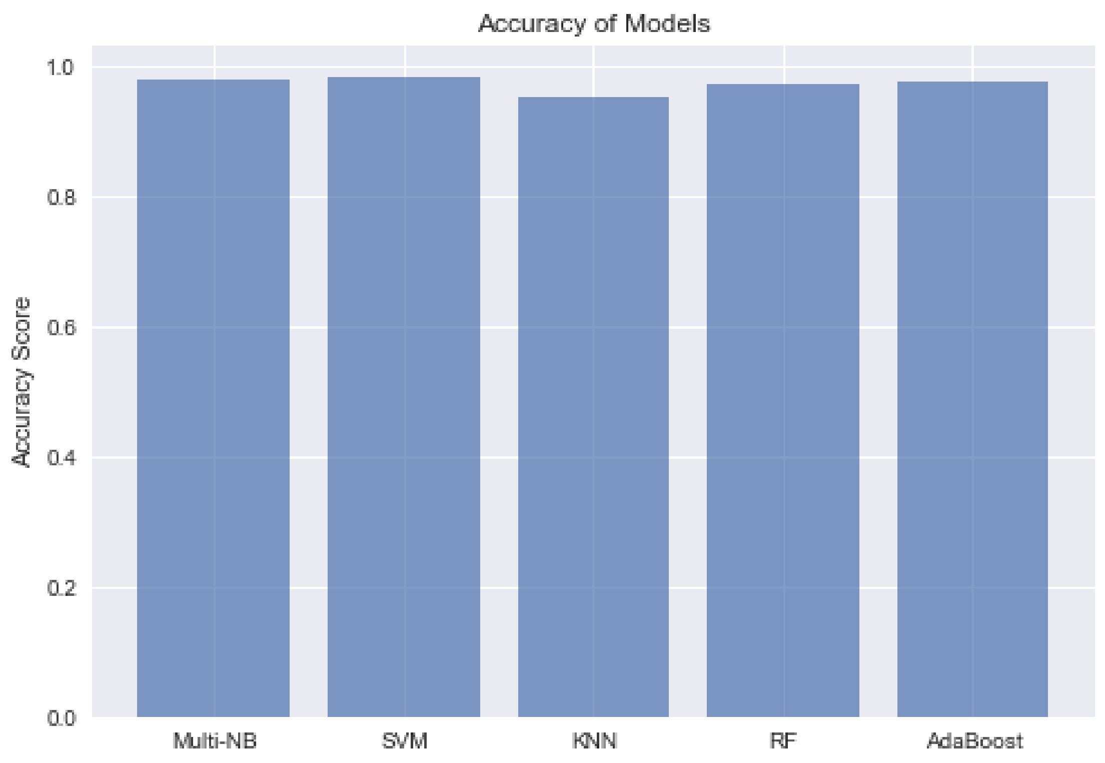
AdaBoost

Accuracy in %:

97.48803827751196

F1 Score

0.8980582524271844



Model	**Research Paper Accuracy (%)**	**Our Implemented Model Accuracy (%)**
Multinomial Naïve Bayes	98.88	97.85
SVM	98.86	98.15
KNN	97.47	95.16
Random Forest	98.57	97.01
AdaBoost	98.59	97.49