CS 7600-Trust Networks

Machine Learning-based SMS Spam Filtering

Swapnika Pasunuri – U00843540

Avinash Reddy Diddakuntla – U00847586

Vinay Mohan Behara – U00851261

1. Abstract:

In this project we extract features from the given data set and use them to train and evaluate different classifiers that are available in the WEKA tool. Using these features, we differentiate spam messages from legitimate messages. In this project, the database of real SMS spams from UCI machine learning repository is used, after pre-processing and feature extraction, different machine learning techniques are applied to the data base. Finally, the results are compared and the best algorithm for spam filtering for text messaging is introduced.

2. Introduction:

These days the ways of generating spam messages are increasing and its nature is diverse. These spam mails are not easily detectable. They are mainly based on the content that includes misspellings, various foreign languages, frequently occurring words and most number of numeric characters. The ham messages on the other hand, have descriptive information without any features above mentioned. They are mostly conversational messages.

3. Procedure:

3.1 Feature Extraction

We generate features from the given dataset using the following parameters:

(i) First Feature

The parameter used to identify the first feature is the number of characters typed in the message. In legitimate messages, we have relatively less number of characters when compared to spam messages. In spam messages they give more information until the it exceeds the limit of SMS. They aim to send sufficient information to customers for illegal benefits. So, the spam messages will be having more numbers of characters than the legitimate ones.

We perform operation by using the predefined function “len” to find the length of the message and print a numeric value for each message.

(ii) Second Feature

The parameter used to identify second feature is the number of currency symbols used in the message. Spam messages use different currency symbols to emphasize the prize or lottery kind of messages. The spam messages are likely to have more currency symbols than the legitimate ones. Mostly used currency symbols are $, £ *etc*.

(iii) Third Feature

In spam messages, we have a lot of numeric strings. They tend to have phone numbers, account details or promotion codes. Generally legitimate SMS messages

don’t include phone number frequently because it is sensitive. So, spam messages contain more numeric strings than legitimate messages.

We perform operations to find the number of numeric strings, even for the alphanumeric words. This helps us get well-built features which results in minimal error.

(iv) Fourth Feature

Generally in legitimate messages we don’t frequently use the same word many times but in spam messages, words like ‘free’,‘won’, ‘cash’ etc are frequently repeated. We find the count of the most frequent word in each message. This value is higher for spam messages compared to legitimate messages.

While building this feature we should not consider the spaces and the punctuations. In each message we calculate the count of each word and print the highest words count.

3.2 Classification

Now, as we have well-built features, we need to know how accurate are these features to detect any new message. We must train the system with data, by which we are making the system learn the pattern of a spam message and a legitimate message.

We are using WEKA to perform the testing by using various classifiers. Each classifier has its own flow by which the accuracy rate or the false positive rate can be recognized.

Here we are using five different Classifiers with the cross validation as 10:

(i) Decision Tree (J48)

Starting from training data we will build a model which is mapped to a tree structure. The goal is to achieve perfect classification with minimal number of decisions. The main challenge while building the tree is to decide on which attribute to split the data at a certain step to have the best split.

This gives the highest accuracy when compared to other classifiers with a good True positive rate.

=== Summary ===

Correctly Classified Instances	5323	95.497 %
Incorrectly Classified Instances	251	4.503 %
Kappa statistic	0.7986	
Mean absolute error	0.0701	
Root mean squared error	0.1944	
Relative absolute error	30.1687 %	
Root relative squared error	57.054 %	
Total Number of Instances	5574	

=== Detailed Accuracy by Class ===

TP rate	FP rate	Precision	Recall	F-measure	MCC	ROC area	PRC area	Class
0.981	0.212	0.968	0.981	0.974	0.800	0.908	0.970	ham
0.788	0.019	0.876	0.788	0.824	0.800	0.908	0.825	spam
0.955	0.186	0.954	0.955	0.954	0.800	0.908	0.950	Weighted Avg.

Table I: Detailed Accuracy by class (Decision Tree)

=== Confusion Matrix ===

```
   a    b   <-- classified as
4734   93 |    a = ham
 158  589 |    b = spam
```

Above result shows the correctly classified instances(accuracy) is 95.497

(ii) Multinomial Naive Bayes:

It is mainly used to calculate the probability that a document belongs to a particular class by applying Bayes theorem. It implements Naive Bayes for multinomially distributed data. The probability for multinomial Naive Bayes is calculated as $P[Ci|D] = (P[D|Ci] \times P[Ci]) / P[D]$
Where 'Ci' is class 'i' and 'D' is the document.

In this section, NB algorithm is applied to the final extracted features. The speed and simplicity along with high accuracy of this algorithm makes it a desirable classifier for spam detection problems.
Results from WEKA:

=== Summary ===

Correctly Classified Instances	5266	94.4743 %
Incorrectly Classified Instances	308	5.5257 %
Kappa statistic	0.7536	
Mean absolute error	0.0888	
Root mean squared error	0.2177	
Relative absolute error	38.2219 %	
Root relative squared error	63.9129 %	
Total Number of Instances	5574	

=== Detailed Accuracy By Class ===

TP rate	FP rate	Precision	Recall	F-measure	MCC	ROC area	PRC area	Class
0.974	0.246	0.962	0.974	0.968	0.754	0.914	0.976	ham
0.754	0.026	0.820	0.754	0.785	0.754	0.914	0.803	spam
0.945	0.217	0.943	0.945	0.944	0.754	0.914	0.953	Weighted Avg

Table II: Detailed Accuracy by class (Multinomial Naïve Bayes)

=== Confusion Matrix ===

a	b	<-- classified as
4703	124	a = ham
184	563	b = spam

The False Positive rate is comparatively higher in this method but has a higher accuracy rate. Improving the Features might give a better rate.

(iii) K-Nearest Neighbours

We can select appropriate value of k based on cross validation. It can also do distance weighing. KNN is the k parameter and it specifies the number of nearest neighbours to use when classifying at a test instance. In this method, the label for a test sample is predicted based on the majority vote of its k nearest neighbors.

=== Summary ===

Correctly Classified Instances	5308	95.2278 %
Incorrectly Classified Instances	266	4.7722 %
Kappa statistic	0.7894	
Mean absolute error	0.0555	
Root mean squared error	0.2153	
Relative absolute error	23.9089 %	
Root relative squared error	63.2065 %	
Total Number of Instances	5574	

=== Detailed Accuracy By Class ===

TP rate	FP rate	Precision	Recall	F-measure	MCC	ROC area	PRC area	Class
0.977	0.206	0.968	0.977	0.973	0.790	0.945	0.989	ham
0.794	0.023	0.841	0.794	0.817	0.790	0.945	0.786	spam
0.952	0.182	0.951	0.952	0.952	0.790	0.945	0.962	Weighted Avg

Table III: Detailed Accuracy by class (K Nearest Neighbours)

=== Confusion Matrix ===

a	b	<-- classified as
4715	112	a = ham
154	593	b = spam

This classifier gives the accuracy rate as 95.22% with the k value as 10. As the value of k changes, the accuracy rate decreases by a small value.

(iv) Support Vector Machines(SVM)

SVM are a classification method that maps class examples to points in space and maximizes the margin around a hyperplane separating the classes. Since the points are not linearly separable, a kernel version can be used to fit the maximum margin in a high dimensional feature space. It transforms nominal attributes to binary ones. It normalizes all attributes by default.

=== Summary ===

Correctly Classified Instances	5275	94.6358 %
Incorrectly Classified Instances	299	5.3642 %
Kappa statistic	0.7422	
Mean absolute error	0.0536	

Root mean squared error 0.2316
Relative absolute error 23.0999 %
Root relative squared error 67.9861 %
Total Number of Instances 5574

=== Detailed Accuracy By Class ===

TP rate	FP rate	Precision	Recall	F-measure	MCC	ROC area	PRC area	Class
0.988	0.323	0.952	0.988	0.970	0.752	0.833	0.951	ham
0.677	0.012	0.897	0.677	0.772	0.752	0.833	0.651	spam
0.946	0.281	0.945	0.946	0.943	0.752	0.833	0.911	Weighted Avg

Table IV: Detailed Accuracy by class (SVM)

=== Confusion Matrix ===

```
   a    b   <-- classified as
 4769   58 |    a = ham
  241  506 |    b = spam
```

The accuracy of SVM is 94.63%, compared to others its accuracy rate is lower.

(v) Random Forest

It is a class for constructing random trees. It is an extension of bagging for decision trees that can be used for classification. The classification is a combination of decision trees built from a bootstrap sample from training set. In this, you can mention the number of estimators to be used, can be 10 or 100.
With this set of features, the classifier gives the accuracy rate as 95.44%
=== Summary ===

Correctly Classified Instances 5320 95.4431 %
Incorrectly Classified Instances 254 4.5569 %
Kappa statistic 0.8007
Mean absolute error 0.0588
Root mean squared error 0.1917
Relative absolute error 25.3145 %
Root relative squared error 56.2847 %
Total Number of Instances 5574

=== Detailed Accuracy By Class ===

TP rate	FP rate	Precision	Recall	F-measure	MCC	ROC area	PRC area	Class
0.976	0.187	0.971	0.976	0.974	0.801	0.944	0.984	ham
0.813	0.024	0.842	0.813	0.827	0.801	0.944	0.864	spam
0.954	0.165	0.954	0.954	0.954	0.801	0.944	0.968	Weighted Avg

Table V: Detailed Accuracy by class (Random Tree)

=== Confusion Matrix ===

```
   a    b   <-- classified as
 4713  114 |    a = ham
  140  607 |    b = spam
```

Random Forest seems the most efficient with appropriate accuracy and False Positive rate.

Final Results of Classifiers applied to SMS Detection Features:

Classifier	Accuracy	True Positive	False Positive
Decision Tree(J48)	95.497%	0.955	0.186
Multinomial Naïve Bayes	94.47%	0.945	0.217
K Nearest Neighbours	95.23%	0.952	0.182
Support Vector Machine	94.64%	0.946	0.281
Random Forest	95.44%	0.954	0.165

4. Conclusion:

The results of multiple classification models applied to the SMS Spam dataset are shown in table VI. From simulation results, Decision Tree(J48) and Random Forest are among the best classifiers for SMS spam detection in terms of Accuracy, True Positive and False Positive. The best Classifier in terms if Accuracy and True Positive is Decision Tress with 95.497%, in terms of False Positive is Random Forest. This values come with the features number of characters, number of currency symbols, number of numeric strings and frequency of the most frequent word.

5. References

(i) SMS Spam Detection Using Machine Learning Techniques - Houshmand Shirani-Mehr,

(ii) ”http://en.wikipedia.org/wiki/Short Message Service”