.Net library for SMS spam detection using machine learning: A cross platform solution

Conference Paper · March 2018

DOI: 10.1109/IBCAST.2018.8312266

1 author:

Syed Sarmad Ali
Beihang University (BUAA)
8 PUBLICATIONS **41** CITATIONS

SEE PROFILE

.Net Library for SMS Spam Detection using Machine Learning

A Cross Platform Solution

Syed Sarmad Ali
School of Computer Science & Engineering
Beihang University (BUAA), Beijing, China.
AND
Dept. of Computer Science & IT
The University of Lahore, Lahore, Pakistan
sarmadali.uol@gmail.com

Junaid Maqsood
Department of Computer Science
Carleton University
Ottawa, Canada
junaid.maqsood@carleton.ca

***Abstract*—Short Message Service is now-days the most used way of communication in the electronic world. While many researches exist on the email spam detection, we haven't had the insight knowledge about the spam done within the SMS's. This might be because the frequency of spam in these short messages is quite low than the emails. This paper presents different ways of analyzing spam for SMS and a new pre-processing way to get the actual dataset of spam messages. This dataset was then used on different algorithm techniques to find the best working algorithm in terms of both accuracy and recall. Random Forest algorithm was then implemented in a real world application library written in C# for cross platform .Net development. This library is capable of using a prebuild model for classifying a new dataset for spam and ham.**

Keywords—spam filter; SMS; detection; machine learning; classification; clustering; algorithms; C# library; online detection

I. INTRODUCTION

Short Message Service (SMS) has indeed occupies the majority of our communication and has become an essential part in daily human activity. According to [1], SMS itself has become a multi-billion industry. It is now a matter of seconds for anyone to connect with others using SMS. With the recent advancements in the technology and with a huge competition among the different cellular companies the cost of sending the SMS has reduced to just about nothing. Now with different cellular packages you get close to unlimited SMS's and the ability to send world-wide and low cost. This along with the betterment has also caused the short message service to be used as a marketing or other un-wanted services. In order to keep the quality of this service in check, proper steps must be taken for the prevention of spam. Spam can be easily described as an unwanted content that is send in a bulk quantity to bulk users. The purpose of spam is to either get users toward a specific marketing scheme or to just scam.

Even today the quantity of spam SMS is quite low than spam emails, but still there is enough quantity to create a miss-leading usage. In 2010-2012, it is reported [2] that about 90% of emails are spam worldwide while this number is very low in terms of SMS. In Asia about 30% of total Messages were actually spam [2]. As the percentage is quite small, there has been more advancements in terms of catching and blocking email-spam but still a very few studies are available for doing the similar thing in terms of SMS.

The recent growth in mobile users because of the recent advancements in smart phones, the popularity of SMS's has increased. This has caused a lot of different communities to create tools and techniques for spamming the user's mobile phones in order to get the desired output. To create a better understanding in terms of machine learning algorithms to sort out the spam and filter them. There existed a lack of a proper dataset as well as a lack on the study for different algorithms and clustering techniques for this specific problem.

In this research paper, a new tool is created based on .Net framework. The resultant tools are actually a cross platform library project which is compatible of using an already normalized dataset to map it within the internal model and to see in real examples what are ham and which are spam.

II. LITERATURE STUDY

In 2013, Houshmand[1] put on dissimilar machine learning algorithms to SMS spam classification problem. Further they analyze and compare the output to achieve the understanding that can sieve the SMS spams. The Author use the database from UCI machine learning repository, explained in [3][4]. An SMS's subset arbitrarily selects ham messages. The dataset constitutes of the label message and trailed by the message string. Methods like SVM and Naïve bayes are imposed to the sample which are initially processed and then features are extracted. Finally, the best classifier will be compared to the dataset discussed in [4]. Matlab was used for feature extraction and the analysis of the data and then different algorithm are applied using the python scikitlearn library.

In 2011, Tiago et al [5] studied this issue and attempted to find different smaller datasets and their own personal study to create a better dataset for academic studies. They created a new collection of 4827 ham and 757 spam SMS and they donated this dataset to the community for further analysis. This was a remarkable step towards finding a solution to stop spam.

In 2012, Coskun and Giura from a research institution in New York City performed an experiment [6] to classify the spam-ham dataset by using the similarity equation. What they did was to create an algorithm capable of performing a block match analysis on a steam of different messages to find a similarity among them. Their hypothesis was that if a lot of messages are similar to previously sent messages than this steam is basically a combination of spam and should be blocked. It was in-fact a smart independent way of classifying spam without using any kind of previous knowledge base. They used an internal algorithm called the Counting Bloom Filter which was capable of finding true similarity. Another interesting thing about their

978-1-5386-3564-3/18/$31.00 ©2018 IEEE

study was that they used YouTube comments to test their algorithm because of their assumption that the comments on videos are quite similar to the Short messages as their number of characters is quite the same and they are just as much filled with spam as messages. The proposed algorithm had a 100% detection rate when a stream of similar messages was flowing. But when less number of messages were matching it had quite less accuracy. In 2013, Shirani-Mehr [7] tried to study the possibility to classify SMS spam using different machine learning algorithms using the UCI spam dataset.

The research was composed of testing five different algorithms on the same exact dataset without any pre-processing. He was able to find that the algorithms will be able to classify the spam dataset much more effectively and he concluded that Multinomial Naïve Bayes was the best among them all while SVM ranked 2nd.

In 2015, Akbari et al [8] did what the 2013 study lacked. They tried to study the different post-processing techniques on the dataset created by the Tiago et al [5] to find if any post-processing technique might be helpful in better classifying the dataset. They tried different wrapper algorithms for better classifications. They founded out that Gentle Boost using R was able to detect the spam out of the complete dataset at an accuracy of 98.3% using Naïve Bayes it resulted in 97.6% detection.

There may exist numerous ways of classifying the SMS spam dataset. Many researches have already studied different algorithms and different boosting filters. In this study we will investigate and put our efforts to formulate pre-processing techniques on dataset. Furthermore, we test the optimized data set on different machine learning algorithms and different clustering techniques, to find if the original dataset after preprocessing had any better results than what previous studies found.

III. GOALS AND OBJECTIVES

This study will investigate the role of data-set as a whole towards the different classification techniques in terms of SMS spam detection. When we discuss about machine learning classification, two main concept exist. First is the dataset and the other algorithm. The dataset is indeed a key factor in making a model capable. If the dataset is not balanced or contains reproductions, then the resultant algorithm will have difficulty performing optimal solution, when given a real world problem. There also subsists the problem that the classifier might become too specific to the dataset making it perform as if it's working well. Mostly that problem is solved by either using 10-fold cross validation or using a different training set and a different testing set. In this study 5-fold cross validation is used because of the time limitation. Comparing to the previous studies Houshmand[1], uses Naïve Bayes algorithm with 10-fold cross validation. Many algorithms take a lot of time in processing the dataset more on that in Section IV.

The pre-processing is the key factor in discussion here. Most of the papers presented previously use the data as is or try only the feature selection approaches. They don't try different string conversion techniques. In this paper the relation of the string conversion to a normalized vector distribution and the result on the performance of the classifiers is discussed. After pre-processing the dataset eight different classifiers are used on it to see the performance.

After having results from different classifiers, T-Test and Fredmen's test will be used to determine if a valid difference exist among the different algorithms and to find which exact algorithm performed statistically better than the others.

Moreover, a new tool library was made based on the best resultant algorithm. This library was constructed in C# language for cross platform .Net development. Due to the new C# 6 launch this library can be used in any platform web based, desktop application, android, iPhone or to be deployed to a live stream server for on the way detection. This library is currently only capable of using an already normalized dataset. Nevertheless, Future work will make it work with the actual text, consequently it can be used to visualize the results just as you type a combination of words. Similarly, for future development more research will be conducted to dynamically change the model based on specific inputs within the C# application. Correspondingly, to add specific functions to enable the choice to algorithm's model and their size.

IV. METHODOLOGY

A. Dataset

The first part of this study was mostly in analyzing the different datasets out there and to do a specific pre-processing technique to see if it will make any difference to the end result of the classifying algorithms. We started with the most generic and the biggest dataset out there [5], presented by Tiago et al to the community. The dataset contained a huge amount of SMS's, but after further analysis the problem in hand was actually a class imbalance problem. There were almost 7 times more instances of ham class than spam. To solve this problem, we decided to study another dataset [7], presented by Dublin Institute of Technology which contained only a collection of 1353 spam messages. After an easy algorithm created in C# language we were able to merge the two datasets and easily format them into a CSV. The end result was a coma separated file containing the class attribute in the first part and the actual text message in the second. It contained 2098 spaminstances and 4808 ham.

B. String to Normatized distribution and Feature Selection

The dataset created had the actual string message which is not that much easy to classify by any automated machine learning algorithm. To further transform it into numeric normalized data WEKA [9] was used. The CSV file was easily converted to an ARFF file which is the most specific file used by the WEKA tool for any kind of classification. To make the dataset a more readable for machine learning algorithms we used StringToWordVector filter which basically maps each combination of words to a vector of numbers each reporting a number instead of a word. We configured the filter, therefore the

resultant data was normalized and contained the number which articulates how many times a specific word was used within a message. After performing this filter, the resultant dataset contained a total of 1047 attributes. At this stage we can introduce a feature selection approach. After using InfoGain Filter with the ranker and the judge the reminder was 353 attributes. These were the ones that were used at least 2 times within the messages and were the ones with the rank and threshold of at least 0.025. After doing so in order to solve the still class imbalance issue we added a few re-sampling filter and gave more weight to the imbalanced class. The resultant dataset had 4419 instances with 353 attributes and was divided as 1944 spam and 2475 ham.

C. Different Classification Algorithms using WEKA

After having a proper dataset, the dataset was then divided into 2 datasets. One was 90% of the original this was called Spam-Weka-Dataset.arff which was used to train and test the classifier using 5-fold cross validation and eight different classification algorithms using WEKA tool. While the other dataset was 10% of the original named
Spam-Library-Dataset.arff, this was made to test the end result library for its actual accuracy. Note that the both datasets did not have any similar values in them meaning they were independent of each other.
Once the datasets were complete. Eight different classifiers were used on them Results in Section V. The list of these classifiers are given below

Naïve Bayes (NB)
Multinomial Naïve Bayes (NB-M)
Support Vector Machine (SVM)
K nearest neighbors (K-NN)
Updatable Naïve Bayes (NB-MU)
Decision Trees (J48)
Cost sensitive Naïve Bayes (NB-C)
Random Forest (RF)

Table 1: List of Algorithms

Three clustering algorithms were also used after un-labelling the dataset, they did not state any significant result.

Farthest First (FF)
Simple K Means (SKM)
X-Means (XM)

Table 2: List of clustering Algorithm

V. RESULT

The dataset created for WEKA was used for classification purpose using different classifiers, eight to be exact. After pre-processing of the dataset Random Forest showed the most promising results in terms of spam detection in SMS. This was done using almost 350 features. SVM and Naïve Bayes also gave good results but they were not the best for this experiment. The first diagram is a bar chart of F-Measures of all eight algorithms. F-Measure is used as it looks at a bigger picture and combines both the precision and recall.

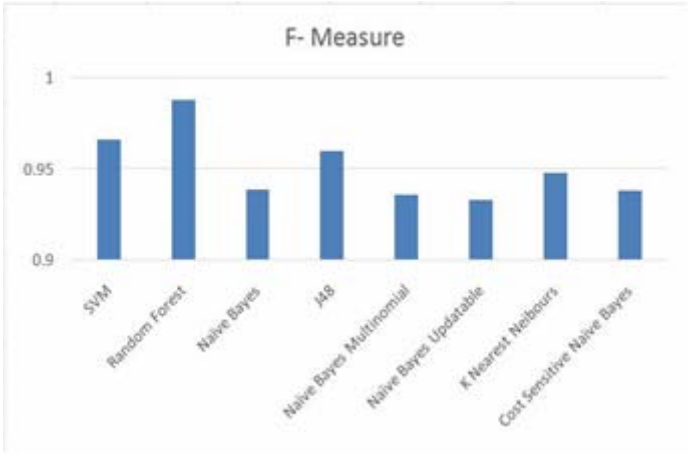

Fig 1: Bar chart of F-Measures of all eight algorithms. F-Measure

The F-Measure clearly show that the Random Forest had a very high margin with the other algorithms.
Houshmand[1] found Naïve Bayes algorithm more accurate as well as speed and accuracy was also good in their analysis.
SVM also did remarkable but other algorithms while retaining speed did not retain accuracy. Random forest was quite slow in terms on model creation and testing but gave good results.

To further test the algorithms many more results were created including the ROC curve. Which we personally prefer because of its neat display and its nature to give a brief overview of the while algorithms working and the level of correctness it has in terms of the dataset in question. The ROC of almost all the algorithms was almost in the same range with the random forest as an exception because it gave a really smooth curve with almost .98 AUC.

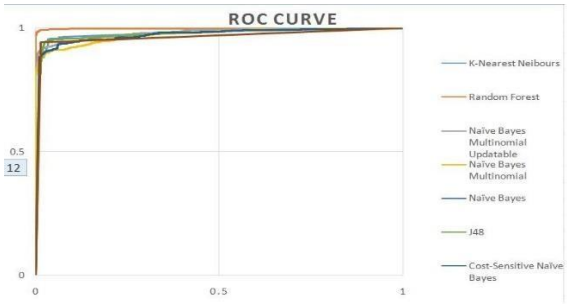

Fig 2: ROC Curve

As it is clearly visible the Random Forest (RF) did have an exceptionally better curve than the other ones. It was studied that this is because of the nature of the dataset it-self being normalized in the way it was, along with the specific features selected.

A. Fredmens Test and the T Test

To better check the accuracy of the algorithms a T test was conducted among them using percentage of correctness as the mean measure and alpha as 5%. The test result showed a significant difference within the different classifiers and also

decision tree was suggested to be the best classifier for this particular data. This is the result of data preprocessed by the two filters previously mentioned

Paired T-Test Two tailed

Dataset	NB	NB-M	NB-MU	SVM	K-NN	NB-C	J48	RF
Main	93.91	93.62	93.35	96.6*	94.8	93.8	96.04*	98.85*

Table 3: T-Test Result *(* means significant difference*)

The SVM and J48 did quite significantly better than the actual baseline which was the Naïve Bayes (NV) but even this test suggested that the Random Forest (RF) had the most percentage of correctness and this was significantly better than the others.

Folds	NB	NB-M	NB-MU	SVM	K-NN	NB-C	J48	RF
1	5	2	1	7	3	4	6	8
2	5	2	1	7	3	4	6	8
3	3	2	4	7	5	1	6	8
4	2	3	4	6	5	1	7	8
5	3	1	2	6	7	4	5	8
R	**18**	**10**	**12**	**33**	**23**	**14**	**30**	**40**

Table 4: Fredmen's Test

Important observation are as follows:
H0: There is no significant difference
H1: There is a significant difference

Chi Square: 27.333
Alpha: 0.05
Decision Rule: 14.067
(Null Hypothesis rejected as Chi Square > Decision Rule

To better test this significance value the Fredmen's test was done among the eight classifiers and RF being ranked 1st in all of the 5 folds made it quite easy for the test to determine the significance of these differences. Our null hypothesis being that there is no difference among any of them was obviously rejected by the Fredmen's test clearly stating that there is a significant difference among all of these. Later on Wilcoxon test proved that the Random Forest (RF) performed the best in terms off-Measure ranking and had the most accuracy and recall value.

Reason of Random Forest Success -Analysis

The reason why The Random Forest classifier was so successful in our experiments can be because as [10] suggested that RF is recognized as an active classifier when dealing with approximations of what variables are significant in the arrangement. RF also equipped with corresponding error in class population disturbed data sets.

DeBarr, D..et al [11] proposed that the reason why Random Forest has been successful is because the strong point of the Random Forest technique comprises feature selection and deliberation of numerous feature subsets.

B. Clustering Algorithms

After testing the different classification approaches another interesting thing would have been to study the different clustering algorithms out there to see if any of them have any accuracy in sorting out the spam without the actual knowledge of their class. To do this the dataset was again converted to a one without the class label. And then the class label was used to study the accuracy of the clusters and to visualize them via color. This same process was done for three clustering algorithms mentioned before and as previously stated they do not give any new results. The reason is because of the nature of this problem. The two classes might not be different in terms of attributes but are more different in terms of patterns among them. These patterns are better found by a classifier than a clustering algorithm.

The first algorithm tested was the SKM which did not found any proper clusters just like the others. It did create two clusters and one was just ham while the other was a mixture of ham and spam. No separate spam cluster was found.

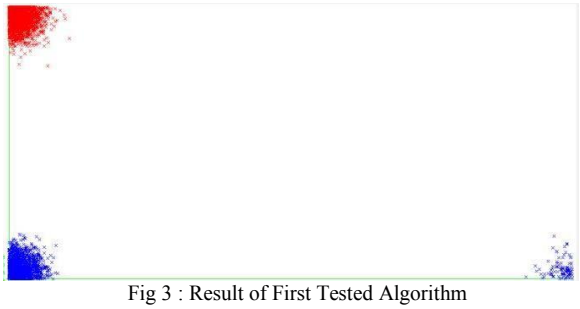

Fig 3 : Result of First Tested Algorithm

The next algorithm was X-Means which also gave the same results as the SKM and did not make any proper findings.

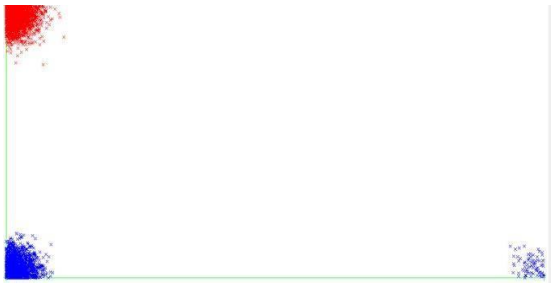

Fig 4: Result of X-Mean Algorithm

The Farthest First on the other hand made both clusters as a combination of both classes.

Fig 5: Result of the Farthest First

This suggested that in terms of clustering there might not be any possibility to determine the data difference between the two classes as they both have similar attributes and they are made from a string attribute. There is no specific difference among

them. It is more like a pattern than differentiate them in terms of their classes. Spam will more likely have a specific pattern or a specific number of words, specific usage of different words like sell, buy, now or some sort of urgency or a sale of a good, as majority usage indicate marketing.

C. Random Forest 100 vs 500 trees

Now that we had clearly found the top accurate algorithm in the spam detection for this particular dataset. The next step was to analyze the actual flow of execution of the Random Forest algorithm on the dataset. For this paper the accuracy, recall and the F-measure was used and different graphs were generated to understand the flow of its execution in terms of instances.

First starting with a generic settings of the random forest and using 100 randomly generated trees from randomly selected nine attributes we got the following flow graphs

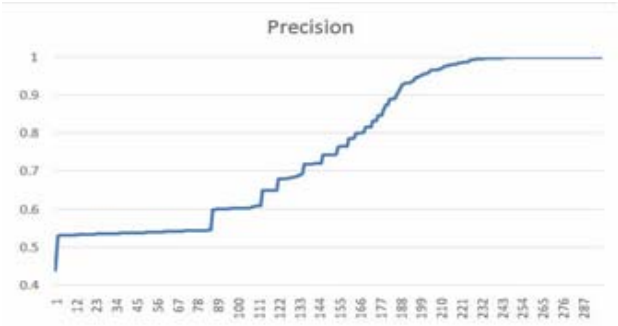

Fig 6: Analysis flow of execution of the Random Forest algorithm

The precision started out slow but after about 80 instances it began to grow and was constantly growing until it reached a constant state.

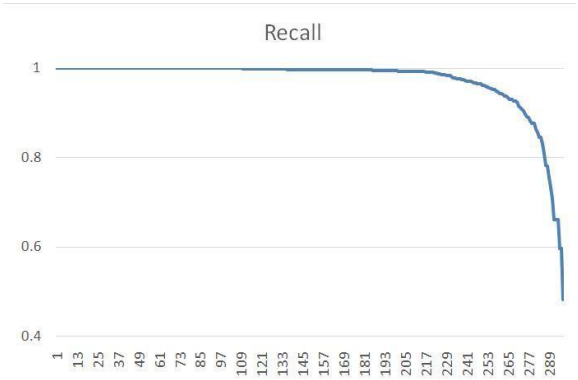

Fig 7: Further Results of the Algorithm

Recall is most of the times the inverse graph than the precision and therefore the graph of recall no surprise was the opposite of the precision, it started on the top and as more instances went through the algorithm it decreased to a certain level.

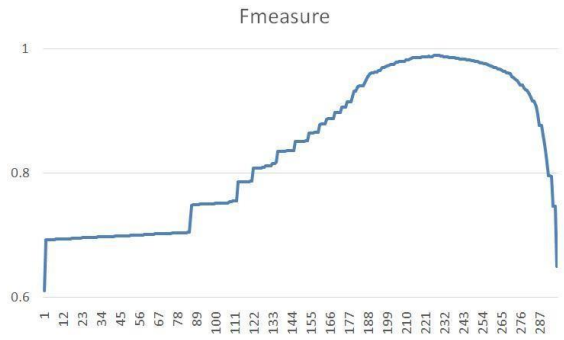

Fig 8: Graph showing the decreased level

The F-measure shows a mix of both precision and recall. It kind of had a zig-zag rise when it starts just like the precision graph but then after reaching a certain level it had a fall similar to the recall. What was interesting to study the same graphs using a different version of random forest generated using 500 trees and nine randomly selected attributes. The difference was when the rise or fall happen but the structure of the graphs remains the same.

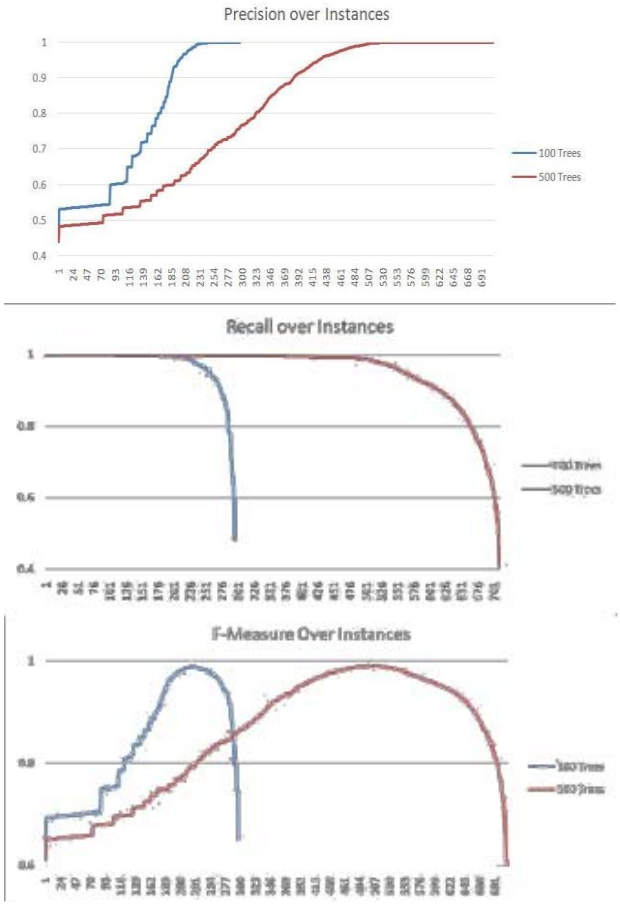

Fig 9: Graph showing the difference in length with previous setting

All three graphs gave the same result as with the previous settings but the difference was the length in the total instances. The 500 tree model was almost 8 times bigger and 5 times slow than the 100 tree model.

After studying the variety of the algorithms and different flavors of Random Forest the RF 100 tree was selected for the next step which is the actual library creation and testing. This will be a remarkable improvement in terms of actual deployment from what was previously done.

VI. .NET LIBRARY

First thing first the actual Random Forest Model was saved using the WEKA tool as a Model file. After this a way to predict un-known instances was determined which used this generated model to classify new instances. As we already had our second dataset which compromised of the 10% of the original, the only thing left was to find a way to use the model within the library.

To use the model, the best choice was to use the WEKA java library. But as it was written and compiled in java language it required a java virtual machine to be used and recompiled. To create this as a .Net module another component called IKVM which is basically a mixed Java and .Net module used to convert a java library file into a .Net dll library. This basically runs the .jar file into a JVM and instead of generating an executable file for windows platform it generates a dll library file for windows platform.

Having a .Net WEKA library was a perk as it can be used among any .Net language. C# 6 was the best choice to create a library that merges this .Net library because it can if wanted be reprocessed into a cross-platform application. The C# language requires a Visual Studio 2015 to be used to make it cross platform as previous versions does not support it

After processing the weka.jar file through IKVM the project needed to include all the various Sun-Java files among the weka.dll file. In total 30 libraries were required by the project to process a model file. The algorithm used within the resultant library is as following:

Algorithm

```
Classifier = SerializationHelper.read(RF.MODEL)
TestDataSet = Instances.read("Spam-Library-Dataset.arff")
TestDataSet.setClassIndex(testDataSet.numAttributes() - 1)
evaluation = new Evaluation(testDataSet)
for all Instance in testDataSet.Instances do
  evaluation.evaluateModelOnceAndRecordPrediction(Classifier, Instance);
end for
for all Prediction in evaluation.predictions do
  if Prediction != null then
    Predicted = prediction.predicted()
    ProcessPrediction(Prediction)
  end if
end for
```

Fig 10: The Algorithm used in the experiment.

In the Process Prediction method, the result can be processed in any way required. In the example project it was just matched with the original class value to test the results.

After some further analysis with the code and the model. It was clear that the model was working well for the dataset. Also a point to note here is that the dataset used is the second divided one. Spam-Library-Dataset.arff which is the one having 10% of the original data. Remember this is that data which the model was not trained on and had no idea about. The evaluation object used in the code basically processes an instance from this dataset to the model and then saves the output class. The class is retuned as a double value in this specific case 0 = spam while 1 = ham. This is because of the arrangement of the attribute values within the ARFF file.

Below is the result computed for all the instances within that 10% dataset computed through the resultant C# 6 library it is stated as a confusion matrix and clearly the performance of decision tree model on this dataset was remarkable. The accuracy for this dataset was 98.64%.

A	**B**	**Classified As**
191	5	**A=Spam**
1	245	**B=Ham**

Table 5: Result computed for all the instance

A. Implication for developer

Our work will bring the question what .NET library has to offer the developers? The answer would be that a single library file, which can then be included simply into any desktop or web based application. Most likely a server which handles communication. Furthermore,

the application is not just for SMS but for any communication platform.

VII. CONCLUSION

The research studied the SMS spam classification problem. The first step was to find multiple datasets which were then combined into one after duplicates removal. The resulting dataset was then processed through StringToWordVector filter to create a normalized distribution of all the strings containing words. This was done without having any stop words. After that feature selection approaches were used to create a more suitable dataset with almost 350 attributes. To solve the class imbalance problem data resampling was done and the data was reduced to a smaller version by favoring the imbalanced class. Then eight different classification techniques were studied on the improved dataset which resulted in Random Forest to have the most accurate classification results. Then the generated model was then processed within an externally created C# library to classify something it had no previous knowledge about. It gave 98.64% result on the dataset it was not trained on through C# library using the Weka model.

VIII. FUTURE WORK

For future work, the library can be improved to make it more adaptable to the environment. So that while on the execution when new messages are received it could classify them and if the predictions are strong it could add that instance to its training set and re-train. This might help it study the new patterns of spam as they emerge. Also the library can be optimized to run on a server while on the go. This will enable for the cellular companies to classify a message before it is being sent. It can also help detect the users that are performing a more number of spams and block them to ensure a better quality service to all other users.

REFERENCES

[1] Mehar, H.S. 2013. SMS Spam Detection using Machine Learning Approach.. International Journal of Information Security Science 2.

[2] Wikipedia-Docs- https://en.wikipedia.org/wiki/Mobile_phone_spamJ. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73

[3] SMS Spam Collection Data Set from UCI Machine Learning Repository, http://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection

[4] SMS Spam Collection v.1, "http://www.dt.fee.unicamp.br/~tiago/ smsspamcollection"

[5] Tiago A. Almeida, Jos Mara G. Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of SMS spam filtering: new collection and results. In Proceedings of the 11th ACM symposium on Document engineering (DocEng '11). ACM, New York, NY, USA, 259-262. DOI=10.1145/2034691.2034742

[6] B. Coskun and P. Giura, "Mitigating sms spam by online detection of repetitive near-duplicate messages," in Communications (ICC), 2012 IEEE International Conference on. IEEE, 2012, pp. 999–1004.

[7] H. Shirani-Mehr, "SMS Spam Detection using Machine Learning Approach",(unpublished)http://cs229.stanford.edu/proj2013/ShiraniMeh r-SMSSpamDetectionUsingMachineLearningApproach.pdf

[8] Akbari, F.; Sajedi, H., "SMS spam detection using selected text features and Boosting Classifiers," in Information and Knowledge Technology (IKT), 2015 7th Conference on , vol., no., pp.1-5, 26-28 May 2015 doi: 10.1109/IKT.2015.7288782.

[9] Sarah Jane Delany, Mark Buckley, and Derek Greene. 2012. SMS spam filtering. Expert Syst. Appl. 39, 10 (August 2012), 9899-9908. DOI=http://dx.doi.org/10.1016/j.eswa.2012.02.053