International Journal of Modern Electronics and Communication Engineering (IJMECE) ISSN: 2321-2152
Volume No.-7, Issue No.-4, July, 2019

An Efficient Scheme for Spammer Identification Using Multiclass Classification in Mobile Cloud

M. Kavitha[1]
Assistant Professor
Information Technology College
CMR Engineering College
Kandlakoya, Medchal, Secunderabad

T. Geetha Lakshmi[2]
Assistant Professor
Computer science and Engineering College
CMR College of Engineering and Technology
Kandlakoya, Medchal, Secunderabad.

Abstract: In machine learning and cyber security fraternity, spam identification methods have been studied for several years. Spammer has been growing since mobile devices usage increases. With the growth of mobile network, spammers have organized into multiple categories for the purpose of benefit maximization, which has caused data Imbalance problem. It is very difficult to categories spammers from multi users owing to the classification of multi-dimensional data. To Avoid this problem, in this paper proposes a Spammer identification Multiclass Classification Scheme based on scikit-learn (SIMCCSL) that utilizes machine learning for industrial mobile networks. We greatly improve the performance of our proposed multi category classification based on scikit-learn. It is a high-level framework designed for supervised and unsupervised machine learning algorithms. Simulation results show that SIMCCSL outperforms these previous schemes in terms of recall, precision, and time complexity.

Keywords: Scikit-learn, Multi-Dimensional, Classification

I. INTRODUCTION

Mobile Networks are becoming popular day by day. As much as Mobile Networks ease the life of human beings, it also gives arise to various problems faced by users using these Mobile Networks. The major problem faced by users using these various Mobile Networks is of Spam. Spam is any unwanted or prohibited behaviour that directly or indirectly violates the certain rules of Mobile Networks. This paper mainly focuses on Spammer identification in Mobile Networks using Spammer identification Multiclass Classification Scheme based on scikit-learn. Mobile Networks has faced various problems due to spam which is created by various spammers to earn and fill their pocket.

The main motive of any spammer is to lure the legitimate user towards his/her malicious spam. According to Mobile Networks Policy, there are various tactics that are considered as Spam:

- Post harmful and malicious links
- Abusive replies to various users
- Posting duplicate and unrelated data for mobile networks
- Posting about current topics for seeking attention

Though, Mobile Networks has followed some security measures to prevent spam but spammers are finding more and more techniques to trap legitimate users. So, basically this paper revolves around the technique of detecting spam in Mobile Networks. By fetching certain live tweets, it will classify the tweets in various content based features and then tell whether a tweet is a spam or not. Multiclass classification is a type of machine learning which has a certain set of data to be put as input onto a certain set of data to be put as output. There can be any number of classes in Multiclass classification and each class is connected to its Nodes.

II. MULTICLASS CLASSIFICATION USING SCIKIT-LEARN

Multiclass classification is a popular problem in supervised machine learning.

Problem – Given a dataset of m training examples, each of which contains information in the form of various features and a label. Each label corresponds to a class, to which the training example belongs to. In multiclass classification, we have a finite set of classes. Each training example also has n features.

For example, in the case of identification of different types of content, "spam", "non-spam" can be features and "human", "hacker" can be different class labels.

In a multiclass classification, we train a classifier using our training data, and use this classifier for classifying new examples

We will use different multiclass classification methods such as, KNN, Decision trees, SVM, etc. We will compare their accuracy on test data. We will perform all this with scikit learn (Python).

Approach –

- Load dataset from source.
- Split the dataset into "training" and "test" data.
- Train Decision tree, SVM , NB, and KNN classifiers on the training data.
- Use the above classifiers to predict labels for the test data.
- Measure accuracy and visualize classification.

Decision tree classifier – Decision tree classifier is a systematic approach for multiclass classification. It poses a set

of questions to the dataset (related to its attributes/features). The decision tree classification algorithm can be visualized on a binary tree. On the root and each of the internal nodes, a question is posed and the data on that node is further split into separate records that have different characteristics. The leaves of the tree refer to the classes in which the dataset is split. In the following code snippet, we train a decision tree classifier in scikit-learn.

Algorithm 1:- Decision Tree Classifier

1) Inputs:- datasets , confusion_matrix, train_test_split
2) # loading the social network dataset
3) socialnetwork = datasets.load_socialnetwork()
4) # loading the socialnetwork dataset
5) socialnetwork = datasets.load_ socialnetwork ()
6) # X -> features, y -> label
7) X = socialnetwork.data
8) y = socialnetwork.target
9) # dividing X, y into train and test data
10) X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)
11) # training a DescisionTreeClassifier
12) load_ DecisionTreeClassifier()
13) dtree_model = DecisionTreeClassifier(max_depth = 2).fit(X_train, y_train)
14) dtree_predictions = dtree_model.predict(X_test)
15) # creating a confusion matrix
16) cm = confusion_matrix(y_test, dtree_predictions)

SVM (Support vector machine) classifier – SVM (Support vector machine) is an efficient classification method when the feature vector is high dimensional. In scikit learn, we can specify the kernel function (here, linear). To know more about kernel functions and SVM refer – Kernel function | scikit learn and SVM.

Algorithm 2:- Svm High Dimensional Data Classification

1) Inputs:- datasets , confusion_matrix, train_test_split
2) # loading the social network dataset
3) socialnetwork = datasets.load_socialnetwork()
4) # X -> features, y -> label
5) X = socialnetwork.data

$$P(y|x_1,...,x_n) = \frac{P(y)P(x_1,...,x_n|y)}{P(x_1,...,x_n)}$$

6) y = socialnetwork.target
7) # dividing X, y into train and test data
8) X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)
9) # training a linear SVM classifier
10) load _SVC ()

$$P(y|x_1,...,x_n) = \frac{P(y)\prod_{i=1}^{n}(P(x_i|y)}{P(x_1,...,x_n))}$$

$$\hat{y} = arg \max_y P(y)\prod_{i=1}^{n}(P(x_i|y)$$

$$P(y|x_1,...,x_n)\alpha(P(y)\prod_{i=1}^{n}(P(x_i|y))$$

11) svm_model_linear = SVC(kernel = 'linear', C = 1).fit(X_train, y_train)
12) svm_predictions = svm_model_linear.predict(X_test)
13) # model accuracy for X_test
14) accuracy = svm_model_linear.score(X_test, y_test)
15) # creating a confusion matrix
16) cm = confusion_matrix(y_test, svm_predictions)

KNN (k-nearest neighbours) classifier – KNN or k-nearest neighbours is the simplest classification algorithm. This classification algorithm does not depend on the structure of the data. Whenever a new example is encountered, its k nearest neighbours from the training data are examined. Distance between two examples can be the euclidean distance between their feature vectors. The majority class among the k nearest neighbours is taken to be the class for the encountered example.

Algorithm 3: KNN High Dimensional Data Classification

1) Inputs:- datasets , confusion_matrix, train_test_split
2) # loading the social network dataset
3) socialnetwork = datasets.load_socialnetwork()
4) # X -> features, y -> label
5) X = socialnetwork.data
6) y = socialnetwork.target
7) # dividing X, y into train and test data
8) X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)
9) # training a KNN classifier
10) load_KNeighborsClassifier ()
11) knn = KNeighborsClassifier(n_neighbors = 7).fit(X_train, y_train)
12) # accuracy on X_test
13) accuracy = knn.score(X_test, y_test)
14) print accuracy
15) # creating a confusion matrix
16) knn_predictions = knn.predict(X_test)
17) cm = confusion_matrix(y_test, knn_predictions)

Naive Bayes classifier – Naive Bayes classification method is based on Bayes' theorem. It is termed as 'Naive' because it assumes independence between every pair of feature in the data. Let (x1, x2, …, xn) be a feature vector and y be the class label corresponding to this feature vector.

Applying Bayes' theorem,

Since, $\mathbf{x_1, x_2, \ldots, x_n}$ are independent of each other,

Inserting proportionality by removing the $\mathbf{P(x_1, \ldots, x_n)}$ (since, it is constant).

Therefore, the class label is decided by,

P(y) is the relative frequency of class label **y** in the training dataset. In case of Gaussian Naive Bayes classifier, $\mathbf{P(x_i \mid y)}$ is calculated as,

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \; exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Algorithm 4:- Naïve Bayes High Dimensional Data Classification

1) Inputs:- datasets , confusion_matrix, train_test_split
2) # loading the social network dataset
3) socialnetwork = datasets.load_socialnetwork()
4) # X -> features, y -> label
5) X = socialnetwork.data
6) y = socialnetwork.target
7) # dividing X, y into train and test data
8) X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)
9) # training a Naive Bayes classifier
10) load_GaussianNB()
11) gnb = GaussianNB().fit(X_train, y_train)
12) gnb_predictions = gnb.predict(X_test)
13) # accuracy on X_test
14) accuracy = gnb.score(X_test, y_test)
15) print accuracy
16) # creating a confusion matrix
17) cm = confusion_matrix(y_test, gnb_predictions)

III. RESULT ANALYSIS

Machine Learning Algorithm	Accuracy Detecting non-spam	Accuracy Detecting Spam	Average Accuracy for both non-spam and Spam
NB	90.71	78.57	85.96
SVM	87.86	77.14	82.96
KNN	92,86	66.71	77.04
Tree	87.86	75.71	82.59

Table:- The Overall Accuracy of the 4 Classifiers

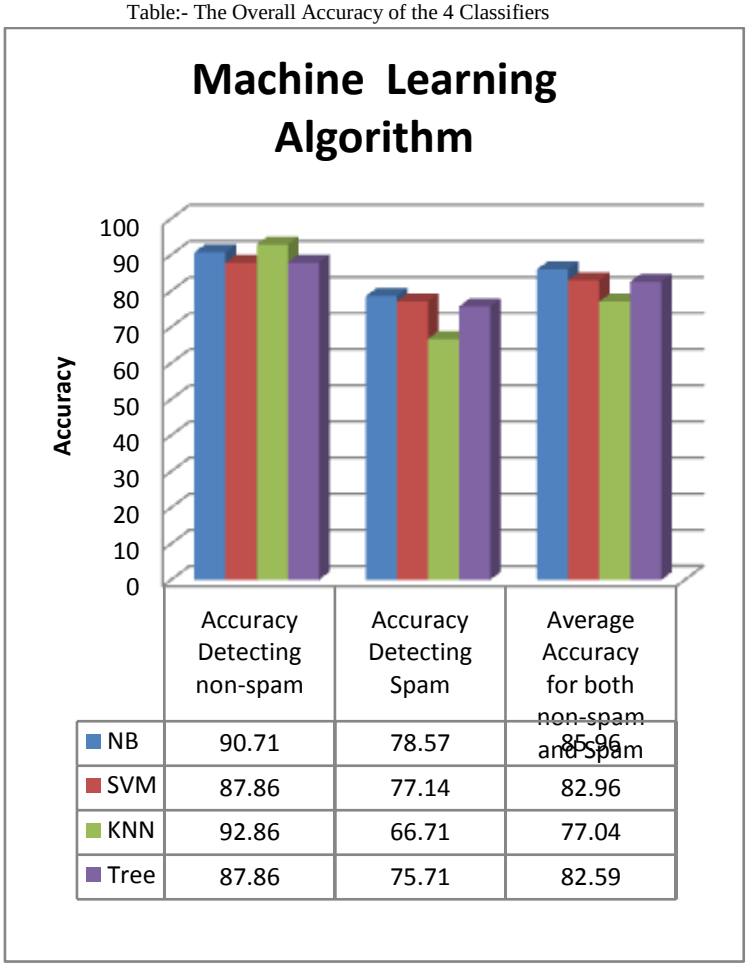

Fig:- The Overall Accuracy of the 4 Classifiers

IV. CONCLUSION

In this research, Social Network has been classified into different categories of spam and non-spam. This research first extracts the social network data and then preprocesses them to refine the data in a similar manner. Then, features are extracted and weights are stored in a file. After performing the Multiclass classification, accuracy, values are calculated this explains that how accurately spam has been classified. Also, this technique can be applied in some other context may be for any other domain. Also, comparison can be performed between various techniques and then results will be calculated.

REFERENCES

[1] N. Krawetz, "Anti-Spam Solutions and Security", The Symantic Blog, (2010) November 2, http://www.symantec.com/connect/articles/anti-spam-solutions-and-security.

[2] D. Mertz, "Spam filtering techniquesSix approaches to eliminating unwanted e-mail", IBM Developer Work Megazine, (2002) September 1, http://www.ibm.com/developerworks/linux/library/l-spamf/index.html.

[3] T. S. Guzella and W. M. Caminhas, "A review of machine learning approaches to Spam filtering", Int Journal of Expert Systems with Applications, vol. 36, (2009), pp. 10206-10222.
[4] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers", ICML 2001 Proceedings of the Eighteenth International Conference on Machine Learning, (2001), pp. 609-616.
[5] P. Pantel and D. Lin, "SpamCop: a spam classification and organization program", In Learning for Text Categorization – Papers from the AAAI Workshop, (1998), pp. 95–98, Madison Wisconsin. AAAI Technical Report WS-98-05.
[6] M. Sahami, S. Dumais, D. Heckerman and E. Horvitz, "A Bayesian approach to filtering junk e-mail", In Learning for Text Categorization – Papers from the AAAI Workshop, (1998), pp. 55–62, Madison Wisconsin. AAAI Technical Report WS-98-05.
[7] P. Graham, "A Plan for Spam", (2002) August, http://paulgraham.com/spam.html.
[8] J. Demsar and B. Zupan, "From Experimental Machine Learning to Interactive Data Mining", White Paper, (2010), http://www.celta.paris-sorbonne.fr/anasem/papers/miscelanea/InteractiveDataMining.pdf.
[9] D. K. Renuka, et al., "Spam Classification Based on Supervised Learning Using Machine Learning Techniques", IEEE 2011 International Conference on Process Automation, Control and Computing (PACC), (2011) July 20-22.
[10] M. W. Berry, J. Kogan and E. P. Jiang, "Content-Based Spam Email Classification using Machine-Learning Algorithms", Chapter 3, In Text Mining: Applications and Theory, Copyright © 2010 John Wiley & Sons, Ltd., (2010).
[11] N. Kang C. Domeniconi and D. Barbar´a, "Categorization and Keyword Identification of Unlabeled Documents", Fifth IEEE International Conference on Data Mining, (2005) November 27-30.
[12] R. E. Madsen, S. Sigurdsson and L. K. Hansen, "Enhanced Context Recognition by Sensitivity Pruned Vocabularies", Proceedings of 17th International Conference on Pattern Recognition (ICPR 2004), vol. 2, (2004), pp. 483-486.
[13] C. Domeniconi and M. Al-Razgan, "Weighted Cluster Ensembles: Methods and Analysis", Technical Report ISE-TR-07-06, Department of Computer Science, George Mason University, (2007) December, http://cs.gmu.edu/~tr-admin/papers/ISE-TR-07-06.pdf
[14] http://scikit-learn.org/stable/modules/naive_bayes.html
[15] https://en.wikipedia.org/wiki/Multiclass_classification
[16] http://scikit-learn.org/stable/documentation.html
[17] http://scikit-learn.org/stable/modules/tree.html.