Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 46 (2015) 841 – 848

Procedia Computer Science

International Conference on Information and Communication Technologies (ICICT 2014)

Investigation of Feature Selection Methods for Android Malware Analysis

Deepa K[a,*], G. Radhamani[b], Vinod .P[c]

[a]*Research and Development centre, Bharathiar University, Coimbatore, India 641 046.*
[b]*Department of IT and Science, Dr.GRD college of Science Coimbatore ,India 641 014,*
[c]*Department of Computer Science and Engineering,SCMS School of Engineering & Technology,Cochin,India 683 582*

Abstract

In this paper we present a method for detecting malicious Android applications using feature selection methods. Three distinguishing features i.e. opcodes, methods and strings are extracted from each Android file and using feature selection techniques, prominent and diverse, top ranking features are mined. Different tree classifiers are used to categorize Android files as either malware or benign. Results show that methods is the most credible feature, which gives accuracy of 88.75% with 600 attributes using Correlation Feature Selection method and Adaboost with J48 as base classifier.

*Keywords:*Android malware; Feature selection methods: Dimensionality reduction;

1. Introduction

Today's smart phones have evolved from mobile phones with more sophisticated functionalities like web browsing, social networking, video viewing, online banking and more. Smart phones are smart due to the operating systems it carries. Android is the most popular one and holds the largest market share[1]. The popularity and also the open source nature have attracted malware writers towards Android[2]. Strict application inspection procedure is not

* Corresponding author. Tel.: +91-9446503448; fax: +9191-484-2450508.
*E-mail address:*deepaharinair@gmail.com

performed before they are published in the Android market. Android security model is based on permission-based mechanisim[4].

Android devices are vulnerable[3] to threats caused by the malicious applications due to the ease of modifiability of the applications. Some of the noticeable vulnerabilities are denial of service attacks that cause requests to be flooded in the servers by the compromised machine, execution of code by the attackers using Android Debugger Bridge (adb), stack based buffer overflow resulting in arbitrary code execution, memory corruption to gain root privileges etc.

The paper aims at developing machine learning based method for Android malware analysis. Recently machine learning techniques are being used to build models for decision making that automatically learns from available data. *Andromaly*[6] uses machine learning techniques to discriminate benign and malignant apps. It is a host based, supervised anomaly detection system, that extracts features to learn about known malware and identifies unknown malware. Experiment was performed on 10 datasets developed from 20 benign games, 20 benign tools and 4 malicious apps. Detection rates were better in case of usage of benign games than benign tools.

Our method extracts static features such as opcodes, methods and strings from apk file. Feature selection methods like Goodman Kruskals, Information Gain and Correlation Feature Selection (CFS) is used to rank each attribute. Finally, a classification model is developed for classifying unseen Android files as malware or benign, using various classifiers like Naïve Bayes, Adaboost (with J48 as base classifier), IBK, Random Forest, J48 and Sequential Minimal Optimization (SMO).

Following are the key contributions of the proposed method: (a) A machine learning based model for classification of malware and benign applications is built using non-signature based method. (b) Usually misclassification occurs due to huge feature lengths, so in our model dimensionality reduction is brought about by using three feature selection methods. (c) The robust model using prominent feature method, achieves accuracy of 88.75% with 600 attributes using Correlation Feature Selection Method with Adaboost classifier.

The remaining sections are organized as follows: Section 2 deals with related works in the field of android malware detection, Section 3 discusses the proposed methodology, Section 4 comprises of experiments and results and Section 5 deals with conclusion.

2. Related Work

Authors in[5] proposed a kernel based behavior analysis of Android malware. It is a signature based detection methodology. To generate signatures, keywords or regular expressions were used. Experiments were performed on 230 applications downloaded from the Android market. The result of this study led to the generations of 16 signatures based on three categories of threats associated with the applications.

In[7], a static mechanism for identifying the malicious code in iCalender by reverse engineering was proposed. Experiment was performed using ApkTool[8], WinZip[17], Dex2Jar[18] and JD-GUI[19]. The resulting code is scrutinized and the malicious code injected in iCalender is determined.

DroidMat, static feature extraction method was proposed by[9]. The experiment was carried out on malware collected from Contagio mobile[14] and benign apps collected from Google Play. The features like permissions, activities, services, etc. inside the manifest file were extracted. Singular value decomposition was applied to find out the most critical features.

Gianluca Dini. Et al, discussed a real-time multilevel anomaly detector that monitors Android both at kernel level and user level, named MADAM[10]. It uses lesser number of features compared to Andromaly[6]. Here machine learning techniques were used to detect unknown malware. After gaining root privilege, a logger module bound inside in the kernel finds out the number of occurrence of system calls. A collector receives features and builds feature vectors. Logger stores it in local files to build training set. A classifier was developed based on this and it decides whether the vectors built are good or malicious.

Survey of current state of mobile malware was discussed in[11]. The authors have analyzed 46 pieces of iOS, Android and Symbian malware that was found in wild since 2009. The same dataset is also used for evaluating the techniques for preventing and identifying mobile malware.

3. Proposed Methodology

In this section we describe our machine learning based model for mobile malware analysis. We have applied feature extraction and dimensionality reduction techniques in our approach. ApkTool[8] is used to disassemble Android files and three predominant features namely opcodes, methods and strings are extracted to develop a model for malware detection. The feature selection methods used were Kruskals method, information Gain and Correlation Feature selection (CFS). The feature vector tables constructed from the selected feature sets were given to the classifier for classification using WEKA[16]. The classifier used were Naïve Byes, Adaboost, Ibk, Random Forest, J48 and Sequential Minimal optimization(SMO) .The proposed framework is shown in Fig 1 and the methodology is described in the following subsection.

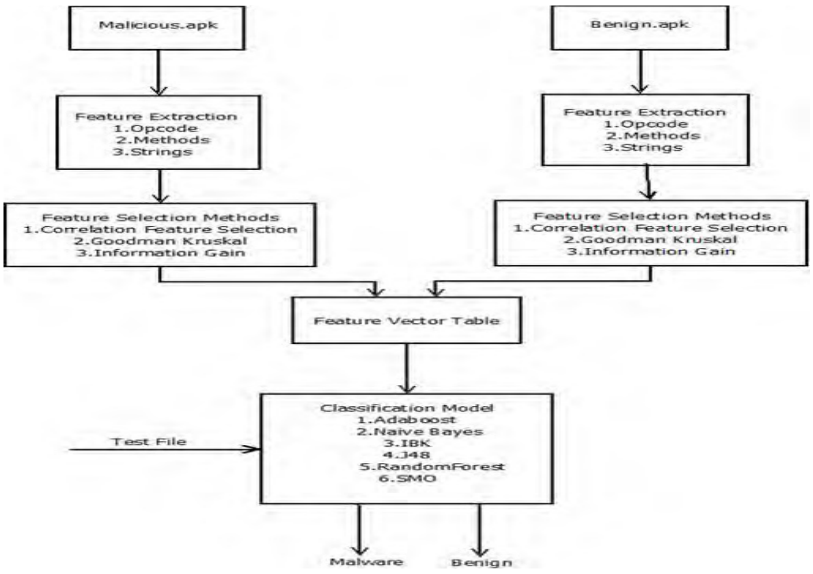
Malicious.apk
Benign.apk
Feature Extraction 1.Opcode 2.Methods 3.Strings
Feature Extraction 1.Opcode 2.Methods 3.Strings
Feature Selection Methods 1.Correlation Feature Selection 2.Goodman Kruskal 3.Information Gain
Feature Selection Methods 1.Correlation Feature Selection 2.Goodman Kruskal 3.Information Gain
Feature Vector Table
Test File
Classification Model 1.Adaboost 2.Naive Bayes 3.IBK 4.J48 5.RandomForest 6.SMO
Malware
Benign

Fig. 1 Architecture of classification model

3.1. Data collection

Datasets were generated for malicious as well as benign Android applications. A total of 612 malicious application samples were collected from the Contagiomobile[14] and 758 benign applications were downloaded from datasets[12] and[13]. Each benign file was scanned using commercial antivirus to make sure that all benign files were indeed legitimate.

3.2. Dataset pre-processing

Android files have .apk extension. These .apk files are given as input to ApkTool[15] to unpackage it. This will contain .smali format. This .smali file is used to extract features, which in our case is opcode (without operands), method name (without return type and parameters) and strings.

3.3. Separation of dataset

After the extraction of features, the complete dataset is divided into train and test set. Out of the total 758 benign samples, 419 of them are included in the training set and remaining reserved for prediction phase. From the total 612 malware samples, 330 are included in the training set and remaining samples are included in the test set.

3.4. Feature selection

A table is constructed for each feature extracted, which contains the frequency of occurrence of the feature and the number of files in which the feature is present, after eliminating sparse and redundant features. Features considered are those which possess low frequency and those exist in fewer samples. Three feature selection methods used are Correlation Feature selection method[20], Goodman Kruskals method[22] and Information Gain method[21]. Table 1 introduces the three feature selection methods.

Table 1 Feature selection methods

Method	**Description**
Correlation Feature Selection Method[20]	$M = \frac{k * r_{fc}}{\sqrt{k + k * (k-1) * r_{ff}}}$ Heuristic based method. Feature is considered good if it has high co-relation to target class and not to other features. M is the heuristic merit of the feature, K the total number of feature, r_{ff} is the average feature to feature co-relation and r_{fc} is the average feature to class co-relation.
Goodman Kruskals Method[22]	$\lambda = \frac{\sum_{i=1}^{r} \max(n_{iP}, n_{iN}) - \max(n_{*P}, n_{*N})}{n - \max(n_{*P}, n_{*N})}$ Used in predicting a class with respect to an input feature. Asymmetric feature λ measures the predectivity. $\lambda = 1/0$ denotes perfect predicitivity/no predictivity. N, total number of samples, i, each feature r, total number of features. P , benign class and N, malware class. n_{ip} / n_{iN} is the number of samples with feature i in class P/N. n_{*p} / n_{*N} is the sum of frequency of samples for all features in class P/N.
Information Gain Method[21]	$IG = -\sum_{K=1}^{C} N_{CK} * \ln \frac{N_{ck}}{N} + \frac{N_F}{N} \sum_{k=1}^{c} \frac{N_{F,ck}}{N_F} \ln \frac{N_{F,ck}}{N_F} + \frac{N_{\overline{F}}}{N} \sum_{k=1}^{c} \frac{N_{\overline{F},ck}}{N_{\overline{F}}} * \ln \frac{N_{\overline{F},ck}}{N_{\overline{F}}}$ IG decides the relevance of an attribute. N, total number of samples, $N_{F,ck} / N_{\overline{F},ck}$ is the number of samples in class ck where feature F is present/absent. $N_F / N_{\overline{F}}$ is the number of samples in which feature F is present/absent. N_F is the number of samples that contain feature F. $N_{ck} / N_{\overline{ck}}$ is the number of samples in class $C_k / \overline{ck}$ ..

3.5. Model construction and prediction

A classification model is constructed with feature lengths 100,200, … 1000 using AdaBoost, Naïve Bayes, Random Forest, Ibk, J48 and Sequential minimal optimization (SMO) classifiers. The models were created and classified using WEKA[16]. The feature vector tables constructed from the selected feature sets are given to the classifier for model construction. The classifier predicts unknown samples.

3.6. Evaluation parameter

The performances of classification using different feature sets were measured in terms of accuracy. Accuracy is the ratio of total number of files that correctly classified, to the number of files in the dataset. It is calculated as follows.

$$Accuracy(Acc) = (TP + TN) / (TP + FN + TN + FP) \tag{1}$$

where, TP is the number of malware files classified as malware, FN is the number of malware files classified as benign, TN is the number of benign files classified as benign and FP is the number of benign files classified as malware. A feature vector table is then constructed with these values.

4. Results

The following tables give the accuracy in classification, based on different feature selection methods. Different portions (100 to 1000) of total features are chosen from the whole feature set in order to determine the effect of feature length on accuracy.

Table 2: Accuracy with Correlation Feature Selection method for feature *method*

Feature Length	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
Classifier										
Adaboost	87.14	86.42	87.5	86.42	87.67	**88.75**	86.96	87.14	87.5	87.67
Naïve Bayes	64.10	68.03	68.39	69.1	69.28	69.1	69.1	69.28	69.46	69.82
Ibk	84.82	83.75	83.21	82.5	81.25	82.32	83.39	84.1	83.21	81.42
J48	85.7	85.53	85.35	85.35	85.71	85.35	85.34	85.53	84.46	83.92
RandomForest	86.1	85.89	85.53	85.89	86.60	86.25	86.07	86.78	86.42	85.71
SMO	69.10	68.75	68.92	68.75	68.03	81.96	81.78	81.96	81.42	81.78

Table 3: Accuracy with Information Gain feature selection method for feature *method*

Feature Length	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
Classifier										
Adaboost	77.68	79.29	79.82	78.57	77.68	77.68	77.68	78.04	78.06	77.32
Naïve Bayes	58.93	58.93	58.93	58.93	58.93	58.93	58.93	58.93	59.11	59.11
Ibk	72.32	72.14	73.21	72.68	72.68	72.68	72.86	72.86	72.86	72.86
J48	74.29	74.29	74.29	74.26	74.29	74.29	74.29	74.29	74.29	74.29
RandomForest	79.29	79.64	78.93	81.25	78.39	81.07	79.82	79.82	79.28	81.61
SMO	56.43	56.42	56.43	58.21	59.64	59.46	59.64	59.29	58.93	59.46

From Table 2, 3 and 4 we observe that higher classification accuracy for feature method is obtained with Adaboost classifier using 600 features extracted from a total feature length of 1000 methods. We also observe that the increase in feature length reduces the classification accuracy due to the participation of irrelevant features in the feature vectors. Worst performance is observed with NaïveBayes classifier. The projected result also depicts that SMO results in higher accuracy at higher feature length.

Table 4: Accuracy with Goodman Kruskal feature selection method for feature *method*

Feature Length	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
Classifier										
Adaboost	83.92	85.18	85	87.5	86.96	85.71	87.32	88.39	87.5	89.11
Naïve Bayes	62.5	64.29	63.03	65	64.82	64.82	64.82	65.71	65.71	65.89
Ibk	78.03	77.5	80	80.71	80	79.64	79.46	79.82	80.54	80.89
J48	82.14	81.43	81.07	83.21	83.39	83.21	83.21	83.21	87.5	86.25
RandomForest	82.5	84.46	85.35	85.36	85.18	85	85.54	85	86.25	86.79
SMO	66.43	65.89	66.61	68.21	68.57	68.39	68.21	68.93	69.82	70.36

Table 5: Accuracy with Correlation Feature Selection method for feature *opcode*

Feature Length	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
Classifier										
Adaboost	87.14	86.43	87.5	86.43	87.68	88.7	86.96	87.14	87.5	87.68
NaïveBayes	64.11	68.04	68.39	69.12	69.29	69.11	69.11	69.29	69.46	69.82
Ibk	84.82	83.75	83.21	82.5	81.25	82.32	83.39	84.11	83.21	81.42
J48	85.7	85.54	85.36	85.36	85.71	85.36	85.35	85.54	84.46	83.93
RandomForest	86.1	85.89	85.54	85.89	86.61	86.25	86.07	86.79	86.42	85.71
SMO	69.10	68.75	68.93	68.75	68.04	81.96	81.79	81.96	81.43	81.79

Table 6: Accuracy with Information Gain feature selection method for feature opcode

Feature Length	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
Classifier										
Adaboost	83.9	82.47	82.47	82.47	82.47	82.47	82.47	82.47	82.47	82.47
Naïve Bayes	56.17	58.14	58.14	58.14	58.14	58.14	58.14	58.14	58.14	58.14
Ibk	81.04	80.68	80.68	80.68	80.68	80.68	80.68	80.68	80.68	80.68
J48	81.75	79.43	79.43	79.43	79.43	79.43	79.43	79.43	79.43	79.43
RandomForest	83.18	84.43	84.43	84.43	84.43	84.43	84.43	84.43	84.43	84.43
SMO	61.72	62.97	62.97	62.97	62.97	62.97	62.97	62.97	62.97	62.97

Table 7: Accuracy with Goodman Kruskal feature selection method for feature *opcode*

Feature Length	100	200	300	400	500	600	700	800	900	1000
Classifier										
Adaboost	83.18	84.44	84.44	84.44	84.44	84.44	84.44	84.44	84.44	84.44
Naïve Bayes	58.5	59.21	59.21	59.21	59.21	59.21	59.21	59.21	59.21	59.21
Ibk	78.83	78.83	78.83	78.83	78.83	78.83	78.83	78.83	78.83	78.83
J48	62.87	62.87	62.87	62.87	62.87	62.87	62.87	62.87	62.87	62.87
RandomForest	83.9	84.26	84.26	84.26	84.26	84.26	84.26	84.26	84.26	84.26
SMO	65.3	62.97	62.97	62.97	62.97	62.97	62.97	62.97	62.97	62.97

From Table 5,6 and 7 we observe that Adaboost and Random Forest classifiers perform well for the feature opcode.

Table 8: Accuracy with Correlation Feature Selection method for feature *strings*

Feature Length	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
Classifier										
Adaboost	62.7	62.7	62.7	62.87	62.87	62.87	62.87	62.87	62.87	62.87
Naïve Bayes	62.56	61.56	61.73	61.73	61.73	61.73	61.73	61.73	61.73	61.73
Ibk	77.85	76.87	76.87	76.87	77.03	77.03	77.04	77.04	77.04	77.04
J48	61.56	61.56	61.56	61.56	61.56	61.56	61.56	61.56	61.56	61.56
RandomForest	79.8	81.59	82.1	82.41	82.08	82.08	82.08	82.08	82.08	82.08
SMO	79.8	79.97	79.97	79.48	79.64	79.64	79.64	79.64	79.64	79.64

Table 9: Accuracy with Information Gain feature selection method for feature *strings*

Feature Length	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
Classifier										
Adaboost	54.56	54.56	54.56	54.56	82.57	82.57	82.57	82.57	82.57	82.57
Naïve Bayes	54.56	54.72	54.72	56.35	66.12	66.12	66.12	66.12	66.12	66.12
Ibk	54.56	54.56	54.56	54.56	81.43	81.43	81.43	81.43	81.43	81.43
J48	54.56	54.56	54.56	54.56	66.12	66.12	66.12	66.12	66.12	66.12
RandomForest	56.51	81.59	58.14	61.07	84.2	84.2	84.2	84.2	84.2	84.2
SMO	56.35	79.97	56.51	59.12	82.57	82.57	82.57	82.57	82.57	82.57

T*a*ble 10: Accuracy with Goodman Kruskal ion feature selection method for feature *strings*

Feature Length	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
Classifier										
Adaboost	63.36	63.36	63.36	63.36	63.36	63.36	63.36	63.36	63.36	63.36
Naïve Bayes	63.19	63.19	63.19	63.19	63.19	63.19	63.19	63.19	63.19	63.19
Ibk	78.83	78.83	78.83	78.83	78.83	78.83	78.83	78.83	78.83	78.83
J48	62.87	62.87	62.87	62.87	62.87	62.87	62.87	62.87	62.87	62.87
RandomForest	82.57	80.62	83.06	83.06	82.57	82.57	82.57	82.57	82.57	82.57
SMO	81.08	81.08	81.6	82.57	81.11	81.11	81.11	81.11	81.11	81.11

The model generated by string as feature does not produce better result. On inspection of feature vector table, we noticed that the prominent list of strings obtained after the application of feature selection algorithm is rarely present in the target class. This means that, the frequency of a feature in a sample are very less, or equal in both classes. Thus the classifier misclassifies the instance. Following are the findings inferred from extensive experiments conducted in this study.

- The developed model using methods resulted in higher accuracy. It was also observed that most of the malware programs utilizes language dependent function instead of user defined procedure/functions.
- Higher accuracy is obtained with boosted version of J48 i.e. AdaBoostM1. The primary reason of better performance is due to bagging and boosting approach in the identification of unseen samples.

- We observe that dimensionality reduction favors machine learning based malware scanner as the elimination of irrelevant attributes generalizes large subset of samples belonging to malware and benign instances. Also, the pruned features have high variance in both the target class.
- Correlation feature selection method produces better result as we have employed two step approaches. In the first step feature to class correlated variables are abstracted and subsequently redundant feature are eliminated by preserving those features that are supposed to have minimal correlation with other attributes.
- On inspection of the vector space representation of the instances using string features, higher misclassification was achieved. We observed that most of the vectors were similar in magnitude hence did not depict effective representation of samples belonging to both class. Moreover, instances had lesser/nearly equal frequencies for majority of the attributes.

5. Conclusion

In this paper we have proposed a static feature based Android malware analysis method using machine learning technique. The system collects prominent features from Android files and uses feature selection methods to identify the top ranking features. The system has calculated the accuracy of each classifier separately. The feature *methods* is having the highest accuracy value of 88.75 when a feature length of 600 is used, compared to all other features. It was found that Adaboost with J48 as base classifier was the best classifier among the six classifiers used. Also among the various feature selection methods, Correlation feature selection method was found to be most accurate. We used the different features independently for testing and classification.

References

1. CNET [Online] (2013, Feb.) http://news.cnet.com/8301-1035_3-57569402-94/android-ios-combine-for-91-percent-of-market
2. McAfee, Santa Clara, CA, USA. (2013). Mcafee Threats Report:Second Quarter 2013.
3. Vulnerabilities: http://www.cvedetails.com/
4. A methology for empirical analysis of permission-based security model
5. Takamasa Isohara ,Keisuke Takemori and Ayumu Kubota.(2011), Kernel-based Behavior Analysis for Android Malware Detection.
6. Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss.(2012) “Andromaly : a behavioral malware detection framework for android devices”, J. Intell. Inf. Syst. 38, 1 February 2012.
7. “icalendar”, [Online], Available: http://www.mediafire.com/?v4c3t2u7zt87eb8
8. APKTool: http://code.google.com/p/android-ap
9. Dong-Jie Wu, Ching-Hao Mao, Te-En Wei, Hahn-Ming Lee1, Kuo-Ping Wu. (2012 “ DroidMat : Android Malware Detection through Manifest and API Calls Tracing ”, In Proceedings of Seventh Asia Joint Conference on Information Security, 2012, 62-70
10. Gianluca Dini1, Fabio Martinelli2, Andrea Saracino, and Daniele Sgandurra.(2012) , “MADAM: a Multi-Level Anomaly Detector for Android Malware”. In the proceedings of 6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, MMM-ACNS 2012, St. Petersburg, Russia, October 17-19, 2012, 240-253
11. Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steven Hanna and David Wagner.(2011), “A Survey of Mobile Malware in the Wild” In the proceedings of SPSM11 , October 17.
12. Benign samples. http://www.appsapk.com/
13. Benign samples. http://www.cell11.com/android/download/apps/
14. Contagiomobile.http://contagiominidump.blogspot.in/2011/07/take-sample-leave-sample-mobile-malware.html
15. APKTool: http://code.google.com/p/android-ap
16. Mark Hall, Eibe Frank,Geofferey Homes, Bernhard Pfahringer,Peter Reutemann,Ian H.Written, The WEKA Data Mining Software:An Update,SIGKDD Exploration ,Volume 11(1),2009
17. WinZip“BriefTutorial-IntroducingWinZip”, http://kb.winzip.com/help/Brief_Tutorial_Introducing_WinZip.htm.
18. “Dex2Jar”, [Online], Available: http://www.cuteandroid.com/tag/dex2jar.
19. “JD-GUI”, [Online], Available: http://java.decompiler.free.fr/?q=jdgui
20. Mark A Hall, Llyod A Smith, Correlation based feature selection for discrete and numeric class machine learning, In the proceedings of 17th International conference on machine learning ICML 2000, pp. 359-366, 2000
21. T.Liu, S.Liu, Zheng Chen, Wei-Ying Ma, An evaluation on feature selection for text clustering, In the Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003
22. Goodman, L.A., Kruskal, W.H, Measures of association for cross classifications, 1954