Classificação de sites phishing

Mestrado em engenharia informática
Mineração de Dados 2021-2022

UNIVERSIDADE DE ÉVORA
ESCOLA DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

Marcelo Feliz, m50356

Trabalho desenvolvido para Mineração de dados, Mestrado em Engenharia Informática.

Évora, Janeiro, 5, 2022

1 Introdução

Phishing, no geral, é uma forma de fraude em que uma pessoa ou múltiplas são alvos contactados por email, mensagem de texto, etc, por um criminoso, estes tentam passar-se por alguém ou entidade legítimo e convencem o indivíduo a dar as suas informações. Neste caso de estudo, ataque cibernético utiliza websites falsos para roubar informação sensível do utilizador como, por exemplo, credenciais de contas, números de cartões de crédito, etc. Pelo mundo, os ataques de phishing continuam a ganhar força e fama. Em junho de 2018, "Anti-Phishing Working Group" (APWG) reportou mais de 51,400 phishing websites únicos. Outro relatório pela RSA estima que organizações globais sofreram perdas de até 9 biliões por causa de incidentes relacionados com phishing. Estas estatísticas provaram que as formas existentes de anti-phishing não são suficientemente eficazes.

A solução mais eficaz que existe de anti-phishing é a "lista negra" do sistema de aviso encontrado em alguns web browsers como o Google Chrome e Mozilla Firefox. A lista negra é composta por sites que se sabem ser phishing, no entanto não conseguem detetar sites lançados recentemente que ainda não estejam na base de dados.

Portanto uma forma que poderá ser mais eficaz e promissora poderá ser utilizando um sistema de machine learning. Esta técnica usa grande número de indicadores conhecidos como características dando assim flexibilidade para reconhecer novos websites de phishing. A eficácia deste método vai cair principalmente sobre as características escolhidas. Para isso, não devemos ter em conta todas as características dos websites, já que muitas características não contribuem para a resolução do nosso problema e iriam apenas deixar o processo mais lento, pesado e com maior utilização de memória, sem trazer qualquer vantagem.

Existem muitos tipos de algoritmos em trabalhos existentes, sendo que neste trabalho vamos avaliar vários algoritmos para avaliar a performance no data set escolhido, e por isso o objetivo passa também por tentar reduzir os atributos da data set que não estejam a ser úteis, mantendo um balanceamento entre eficácia e precisão.

A estratégia deverá passar por examinar as características visuais para tentar detetar anormalidades. Vários algoritmos serão usados para tentar perceber qual o melhor e mais eficaz na previsão

2 Data Set

Este data set contém 48 características extraídas de 5000 paginas web de phishing e 5000 paginas web legítimas, que foram baixadas desde janeiro ate Maio de 2015 e de Maio ate Janeiro de 2017. As características dos sites foram depois processadas. Este data set está preparado para ser utilizado pelo WEKA. Os atributos a seguir apresentados encontram-se descritos na tabela 1. Este data set vai ser referido como sendo o data set 1.

```
@attribute NumDots numeric
@attribute SubdomainLevel numeric
@attribute PathLevel numeric
@attribute UrlLength numeric
@attribute NumDash numeric
@attribute NumDashInHostname numeric
@attribute AtSymbol numeric
@attribute TildeSymbol numeric
@attribute NumUnderscore numeric
@attribute NumPercent numeric
@attribute NumQueryComponents numeric
@attribute NumAmpersand numeric
@attribute NumHash numeric
@attribute NumNumericChars numeric
@attribute NoHttps numeric
@attribute RandomString numeric
@attribute IpAddress numeric
@attribute DomainInSubdomains numeric
@attribute DomainInPaths numeric
@attribute HttpsInHostname numeric
@attribute HostnameLength numeric
@attribute PathLength numeric
@attribute QueryLength numeric
@attribute DoubleSlashInPath numeric
@attribute NumSensitiveWords numeric
@attribute EmbeddedBrandName numeric
@attribute PctExtHyperlinks numeric
@attribute PctExtResourceUrls numeric
@attribute ExtFavicon numeric
@attribute InsecureForms numeric
@attribute RelativeFormAction numeric
@attribute ExtFormAction numeric
@attribute AbnormalFormAction numeric
@attribute PctNullSelfRedirectHyperlinks numeric
@attribute FrequentDomainNameMismatch numeric
@attribute FakeLinkInStatusBar numeric
@attribute RightClickDisabled numeric
@attribute PopUpWindow numeric
@attribute SubmitInfoToEmail numeric
@attribute IframeOrFrame numeric
@attribute MissingTitle numeric
@attribute ImagesOnlyInForm numeric
@attribute SubdomainLevelRT numeric
@attribute UrlLengthRT numeric
@attribute PctExtResourceUrlsRT numeric
@attribute AbnormalExtFormActionR numeric
@attribute ExtMetaScriptLinkRT numeric
@attribute PctExtNullSelfRedirectHyperlinksRT numeric
@attribute CLASS_LABEL {0,1}
```

3 Metodologia

Inicialmente, iremos aplicar vários algoritmos como, ZeroR, J48, random forest, bagging e SMO ao dataset com todos os atributos, faremos o mesmo depois de termos alguns atributos eliminados. Teremos, por isso, em conta a percentagem de previsões corretas assim como o tempo de execução de cada algoritmo.

Teremos o cuidado de ter os testes com um bom balanceamento entre classes

Dado que o data set tem 48 atributos a primeira coisa que será feita é a verificação da utilidade de cada um desses atributos. Como sabemos, quantos mais atributos mais complexo é o algoritmo e, consequentemente, mais demorado de executar. Como este tem como objetivo prever se o site é legitimo ou usado para enganar utilizadores, é importante que o algoritmo seja leve e rápido a executar, no entanto não deve perder a sua credibilidade e eficácia. O algoritmo utilizado é o seguinte:

CfsSubsetEval :

Avalia o valor de um subconjunto de atributos considerando a capacidade preditiva individual de cada recurso juntamente com o grau de redundância entre eles. Subconjuntos de recursos que são altamente correlacionados com a classe, embora tenham baixa intercorrelação, são preferidos.

OPÇÔES:
localmentePredictive – Identifica atributos preditivos localmente. Adiciona iterativamente atributos com a maior correlação com a classe, desde que ainda não haja um atributo no subconjunto que tenha uma correlação mais alta com o atributo em questão.

missingSeparate – Trata falta como um valor separado. Caso contrário, as contagens de valores ausentes são distribuídas entre outros valores na proporção de sua frequência.

Metodo de pesquisa; attributeSelection.BestFirst:

Pesquisa o espaço de subconjuntos de atributos por escalada gananciosa aumentada com um recurso de retrocesso. Definir o número de nós consecutivos sem melhoria permitidos controla o nível de retrocesso realizado. bestfirst pode começar com o conjunto vazio de atributos e pesquisar para frente, ou começar com o conjunto completo de atributos e pesquisar para trás, ou começar em qualquer ponto e pesquisar em ambas as direções (considerando todas as possíveis adições e exclusões de atributos únicos em um determinado ponto).

OPÇÔES:
direction – "FORWARD" seleciona a direção da procura

lookupCacheSize – "1" Defina o tamanho máximo do cache de pesquisa dos subconjuntos avaliados. Isso é expresso como um multiplicador do número de atributos no conjunto de dados. (padrão = 1).

searchTermination – "5" Especifique o número de nós consecutivos sem melhoria a serem permitidos antes de encerrar a pesquisa.

4 Resultados e Discussão

Em linha com o objectivo do trabalho deu-se importância ao tempo que demora a criar o modelo, o tempo de classificação é desprezado, porque em ambiente real o algoritmo irá classificar apenas o site que o utilizador entrou. Usando todos os atributos testou-se alguns algoritmos com várias definições e obtivemos os seguintes resultados:

ZeroR
50%
Obs.
classificou todas como phishing,(apenas como curiosidade).

J48
97.31%
tempo necessário para criar o modelo foi 0.63 segundos
Obs.
classificando 130 errado como não sendo phishing e 139 como sendo phishing quando não eram.

trees.RandomForest
98.31%
tempo necessário para criar o modelo foi 4.27 segundos
Obs.
classificando 88 errado como não sendo phishing e 81 como sendo phishing quando não eram

Bagging
97.38%
tempo necessário para criar o modelo foi 1.72 segundos
Obs.
Classificou 119 sites phishing como legítimos e 143 o contrário

SMO
93.87%
tempo necessário para criar o modelo foi 5.97 segundos
Obs.
Classificou 265 sites phishing como legítimos e 348 o contrario

Discução

Com a utilização destes algoritmos foi possível perceber que RandomForest têm a maior percentagem de sites corretamente classificados, no entanto, apesar de não ser o que demorou mais tempo, demorou bastante tempo comparado com o J48, que obteve uma pontuação de 97.31% (um pouco pior) no entanto demorou apenas 0.63 segundos a criar o modelo.

5 Redução de atributos

Como temos 48 atributos é provável que nem todos estejam a ser úteis, portanto às vezes pode ser bom reduzir o tamanho do Data set com pouca ou nenhuma perda de eficácia quando o ganho na velocidade de processamento e necessidade de memória diminuem drasticamente.

Depois de reduzir os atributos para 13, utilizando o CfsSubsetEva como avaliador do atributo e bestfirst como método de procura passámos a utilizar:

```
attributos: 1,3,5,14,25,27,30,34,35,38,39,47,48;
     NumDots
     PathLevel
     NumDash
     NumNumericChars
     NumSensitiveWords
     PctExtHyperlinks
     InsecureForms
     PctNullSelfRedirectHyperlinks
     FrequentDomainNameMismatch
     PopUpWindow
     SubmitInfoToEmail
     ExtMetaScriptLinkRT
     PctExtNullSelfRedirectHyperlinksRT
```

Os atributos estão sem ordem específica, e mostram que o mais importante na deteção de phishing é: (1) número de pontos no URL, (3) a profundidade do caminho para o website, (5) o numero de "-" no URL, (14) número de caracteres numéricos, (25) o número de palavras sensíveis como (“secure”, “account”, “webscr”, “login”,“ebayisapi”, “signin”, “banking”, “confirm”) no URL, (27) percentagem de links externos no código HTML, (30) existência de protocolo HTTPS, (34) a percentagem de hyperlinks que redirecionam para a própria página, "" ou "file://E:/", (35) o nome de domínio mais frequente no HTML é ou não o mesmo do URL do site, (38) pop-ups de javascript, (39) existência da função "mailto" no HTML, (47) percentagem de meta, script e link tags que contêm links externos nos atributos, (48) percentagem de hyperlinks no HTML que usam domínios diferentes, começam por "" ou usam "javascript::void(0)".

Podemos ver que aquilo que normalmente se acha que tem mais importância na verdade não tem tanta, como por exemplo, número de pontos no link ou existir HTTPS no URL, isto pode ser fruto da evolução dos criadores dos mesmos, que acabam por se focar nos sinais mais importantes para as pessoas considerarem o seu site seguro, e portanto temos de ter atenção que as regras estão constantemente a mudar e será sempre possível com a certa dedicação criar um site phishing que aparente ser o mais real possível, no entanto a grande maioria é detetável.

A figura 1 e 2 mostram a relação entre alguns destes atributos, e pelo gráfico podemos perceber as características que normalmente um site legítimo e um site de phishing possuem.

6 Resultados e discussão depois da redução

Depois da redução, e agora com apenas 13 atributos, utilizámos novamente os mesmos atributos com as mesmas definições e obteve-se os seguintes resultados:

J48
97.31% -> 96.72%
tempo necessário para criar o modelo foi 0.3 segundos
0.63 -> 0.3 seg

trees.RandomForest
98.31% - > 97.43%
tempo necessário para criar o modelo foi 2.68 segundos
4.27 - > 2.68 seg

Bagging
97.38% -> 96.75%
tempo necessário para criar o modelo foi 0.5 segundos
1.72 -> 0.5 seg

Discussão

Como era de se esperar os tempos dos algoritmos diminuíram drasticamente, não só isso mas também em termos de memória, deverá ser utilizado agora apenas 27%(- 73% de memoria) com 13 atributos do que o anteriormente com 48, no entanto tivemos uma pequena perda na classificação.

7 Conclusão

Com a conclusão deste trabalho podemos perceber que sites phishing podem ser detetados com uma eficácia consideravelmente boa de até 98.31%. Com a redução de atributos podemos reduzir consideravelmente o tempo de execução do algoritmo e também a memória necessária para armazenar os dados(-73% do necessário), o que pode ser muito útil para vários utilizadores. No entanto como sabemos, cada utilizador tem a sua preferência e necessidade, e portanto não podemos dizer qual a melhor opção para cada um.

Assumindo que o programa iria ser executado na forma de plug in no navegador, uma solução seria equipá-lo de definições, que os vários utilizadores poderiam facilmente utilizar e obter uma proteção adicional. Isto significa que, por exemplo, um utilizador que não necessite de poupar memória ou que dá o máximo de importância à segurança podia ter como definição a utilização da primeira versão do algoritmo RandomForest com 48 atributos. No entanto outro utilizador que não quer ou não pode gastar tanta memória podia seleccionar outra opção que o levaria a utilizar Bagging, e ainda, se quisesse que fosse mais rápido então a utilização do J48 seria a ideal, estas últimas deveriam ser aliadas ao RandomForest sempre que o utilizador quissese ter segurança adicional num site especifico, com ou sem os 48 atributos, dependendo da existência deles no sistema do utilizador, utilizando-os neste caso se os possuir.

Em suma, a utilização de qualquer um destes algoritmos irá providenciar uma boa defesa contra quase todos os sites de phishing atualmente. Em utilização, no entanto os dados deveriam ser atualizados regularmente, e tendo em conta a possibilidade de alteração da tecnologia, a possibilidade de criação de novas versões com atributos diferentes.

8 Anexos

Tabela 1

No	Identifier	Value type	Description
1	NumDots	Discrete	Counts the number of dots in webpage URL.
2	SubdomainLevel	Discrete	Counts the level of subdomain in webpage URL.
3	PathLevel	Discrete	Counts the depth of the path in webpage URL.
4	UrlLength	Discrete	Counts the total characters in the webpage URL.
5	NumDash	Discrete	Counts the number of "-" in webpage URL.
6	NumDashInHostname	Discrete	Counts the number of "-" in hostname part of webpage URL.
7	AtSymbol	Binary	Checks if "@" symbol exist in webpage URL.
8	TildeSymbol	Binary	Checks if " " symbol exist in webpage URL.
9	NumUnderscore	Discrete	Counts the number of "$"$inwebpageURL.
10	NumPercent	Discrete	Counts the number of "%" in webpage URL.
11	NumQueryComponents	Discrete	Counts the number of query parts in webpage URL.
12	NumAmpersand	Discrete	Counts the number of "&" in webpage URL.
13	NumHash	Discrete	Counts the number of "" in webpage URL.
14	NumNumericChars	Discrete	Counts the number of numeric characters in the webpage URL.
15	NoHttps	Binary	Checks if HTTPS exist in webpage URL.
16	RandomString	Binary	Checks if random strings exist in webpage URL.
17	IpAddress	Binary	Checks if IP address is used in hostname part of webpage URL.
18	DomainInSubdomains	Binary	Checks if TLD or ccTLD is used as part of subdomain in webpage URL.
19	DomainInPaths	Binary	Checks if TLD or ccTLD is used in the path of webpage URL.
20	HttpsInHostname	Binary	Checks if HTTPS in obfuscated in hostname part of webpage URL.
21	HostnameLength	Discrete	Counts the total characters in hostname part of webpage URL.
22	PathLength	Discrete	Counts the total characters in path of webpage URL.
23	QueryLength	Discrete	Counts the total characters in query part of webpage URL.
24	DoubleSlashInPath	Binary	Checks if "//" exist in the path of webpage URL.
25	NumSensitiveWords	Discrete	Counts the number of sensitive words (i.e., "secure", "account", "webscr", "login", "ebayisapi", "signin", "banking", "confirm") in webpage URL.
26	EmbeddedBrandName	Binary	Checks if brand name appears in subdomains and path of webpage URL [30]. Brand name here is assumed as the most frequent domain name in the webpage HTML content.
27	PctExtHyperlinks	Continuous	Counts the percentage of external hyperlinks in webpage HTML source code .
28	PctExtResourceUrls	Continuous	Counts the percentage of external resource URLs in webpage HTML source code .
29	ExtFavicon	Binary	Checks if the favicon is loaded from a domain name that is different from the webpage URL domain name .

30	InsecureForms	Binary	Checks if the form action attribute contains a URL without HTTPS protocol .
31	RelativeFormAction	Binary	Checks if the form action attribute contains a relative URL.
32	ExtFormAction	Binary	Checks if the form action attribute contains a URL from an external domain .
33	AbnormalFormAction	Categorical	Check if the form action attribute contains a “”, “about:blank”, an empty string, or “javascript:true”.
34	PctNullSelfRedirectHyperlinks	Continuous	Counts the percentage of hyperlinks fields containing empty value, self-redirect value such as “”, the URL of current webpage, or some abnormal value such as “file://E:/”.
35	FrequentDomainNameMismatch	Binary	Checks if the most frequent domain name in HTML source code does not match the webpage URL domain name.
36	FakeLinkInStatusBar	Binary	Checks if HTML source code contains JavaScript command onMouseOver to display a fake URL in the status bar.
37	RightClickDisabled	Binary	Checks if HTML source code contains JavaScript command to disable right click function.
38	PopUpWindow	Binary	Checks if HTML source code contains JavaScript command to launch pop-ups .
39	SubmitInfoToEmail	Binary	Check if HTML source code contains the HTML “mailto” function.
40	IframeOrFrame	Binary	Checks if iframe or frame is used in HTML source code.
41	MissingTitle	Binary	Checks if the title tag is empty in HTML source code.
42	ImagesOnlyInForm	Binary	Checks if the form scope in HTML source code contains no text at all but images only.
43	SubdomainLevelRT	Categorical	Counts the number of dots in hostname part of webpage URL. Apply rules and thresholds to generate value.
44	UrlLengthRT	Categorical	Counts the total characters in the webpage URL. Apply rules and thresholds to generate value.
45	PctExtResourceUrlsRT	Categorical	Counts the percentage of external resource URLs in webpage HTML source code. Apply rules and thresholds to generate value.
46	AbnormalExtFormActionR	Categorical	Check if the form action attribute contains a foreign domain, “about:blank” or an empty string. Apply rules to generate value.
47	ExtMetaScriptLinkRT	Categorical	Counts percentage of meta, script and link tags containing external URL in the attributes. Apply rules and thresholds to generate value.
48	PctExtNullSelfRedirectHyperlinksRT	Categorical	Counts the percentage of hyperlinks in HTML source code that uses different domain names, starts with “”, or using “JavaScript ::void(0)”. Apply rules and thresholds to generate value.

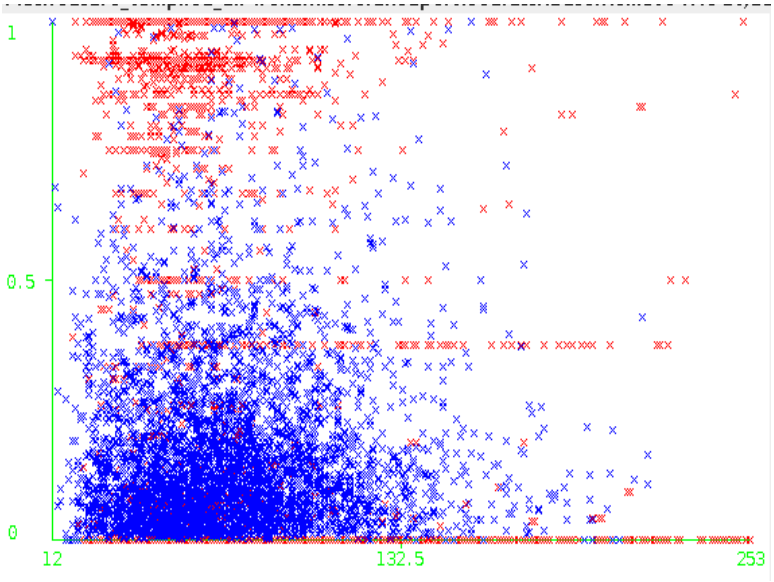

Figure 1: X:(X),Y:(27)

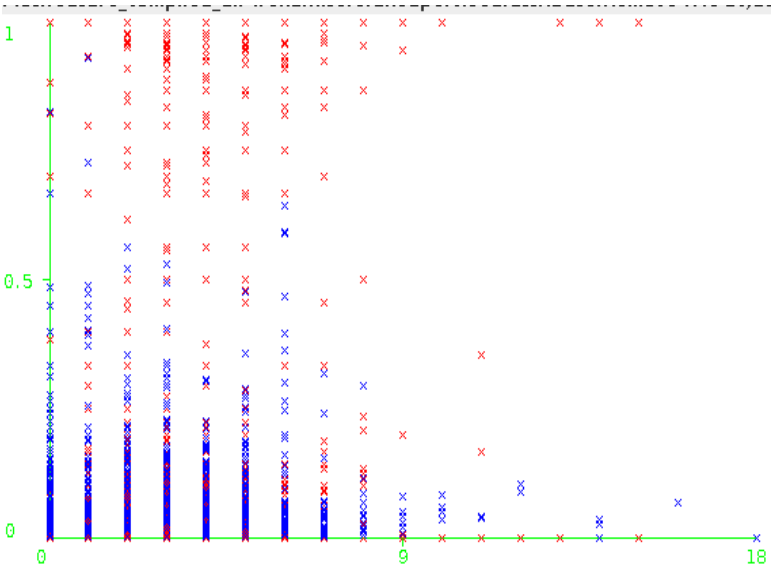

Figure 2: X:(3),Y:(34)

9 Referencias

1. What is phishing | Attack techniques scam examples | Imperva, Imperva, 2016. [Online]. Available: https://www.imperva.com/learn/application-security/phishing-attack- scam/. [Accessed: 12- Jun- 2019]

2. Decision Trees — scikit-learn 0.22.1 documentation, Scikit-learn.org. [Online]. Avail- able: https://scikit-learn.org/stable/modules/tree.html. [Accessed: 10- Jun- 2019]

3. Support Vector Machines — scikit-learn 0.22.1 documentation, Scikit-learn.org. [Online]. Available: https://scikit-learn.org/stable/modules/svm.html. [Accessed: 10- Jun- 2019]

4. Ibrahim, D., Hadi, A. (2017). Phishing Websites Prediction Using Classification Techniques. International Conference on New Trends in Computing Sciences (ICTCS). DOI: 10.1109/ictcs.2017.38

5. Abdelhamid, N. (2016). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. Available: http://archive.ics.uci.edu/ml/datasets/Website+Phishing. [Accessed: 16- Jun- 2019]

6. Chiew, K. L., Tan, C. L., Wong, K., Yong, K., Tiong, W. K. (2019). A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. I*nformation Sciences, 484*, 153-166.

7. Khan, S. A., Hussain, A. (s.d.). Phishing Attacks and Websites Classification Using Machine Learning and Multiple Datasets (A Comparative Analysis). University of Sheffield, UK