Relatório de Ciência de Dados (CI1030) - Vetor de características e Distribuição do conjunto de dados

Nome: Christian Debovi Paim Oliveira
GRR: GRR20186713
Data de entrega: 24/03/2022

1. Introdução

Este relatório visa apresentar as informações obtidas através da exploração de um dataset de informações léxicas de Urls benignas e maliciosas [1] , que é alvo de estudo do paper “Detecting Malicious URLs Using Lexical Analysis” [2]. Foram realizadas avaliações do dataset para a escolha de atributos para a formação de um vetor de características para classificação de cada exemplo fornecido. Também foi proporcionado uma representação gráfica número de exemplo de cada classe do dataset.

2. Descrição do dataset

Este dataset contém arquivos texto contendo 165.366 urls, as quais são divididas em diferentes classes:

- **Benign:** Domínios coletados da lista de top sites da Alexa, onde foi usado um web crawler para extrair as URLs dos domínios. Foi utilizado o serviço do VirusTotal para classificá-las como benignas.
- **Spam:** URLs spam coletadas do dataset público WEBSPAM-UK2007.
- **Phishing:** URLs obtidas do OpenPhish, um repositório ativo de sites phishing.
- **Malware:** URLs contendo ou com relação a malwares, obtidas do projeto DNS-BH, que mantém uma lista de sites com malware.
- **Defacement:** URLs extraídas das páginas da lista Alexa de sites confiáveis, que são URLs ocultas ou forjadas que contém páginas web maliciosas.

Foram extraídas características de 36.707 urls aleatórias de todas as classes (mais detalhes na seção **). No total, foram definidas 79 características léxicas (como tamanho da url, número de tokens/palavras no domínio/path/tld da url, entre outros) para serem extraídas de cada url, que são descritas em detalhes na seção 3.1 do paper [2]. Os resultados da extração foram armazenados num arquivo CSV com 80 colunas, onde a última coluna é o rótulo da classe.

3. Experimento

3.1 Seleção dos atributos

Para verificar a relação entre o valor dos atributos, foi feito uma scatter_matrix com o módulo pandas em python com alguns grupos de atributos para verificar se existia algum tipo de agrupamento/clustering nos dados comparando as características dois a dois. Porém, os gráficos não mostraram diferenças visuais claras entre as classes. Como não seria prático a verificação visual em três dimensões de um total de 79x79 gráficos, foi utilizada outra abordagem para selecionar os atributos.

Assim, os atributos selecionados para o vetor de características foram escolhidos com base na estratégia no paper [2], que selecionou as atributos mais discriminantes através de dois algoritmos de seleção/teste de atributos de machine learning, CfsSubsetEval e Infogain (Ganho de informação). No caso da classificação multi-classe, subentende-se pela Tabela 1 do paper [2] que foi utilizado somente o Ganho de informação, juntamente com um "Ranker" que faz a avaliação individual de cada atributo.

O algoritmo de Ganho de Informação é um dos algoritmos que mede a redução da entropia (que seria a medida de aleatoriedade ou impredictibilidade da característica) causada pela partição dos exemplos de acordo com os valores do atributo.

No final, os atributos escolhidos foram 12 dos 79, os quais estão listados abaixo. A justificativa de usar proporções ou tamanhos dos campos da url se dá pela ideia de que, usualmente, urls maliciosas apresentarem tamanhos mais variáveis/anormais em seus tokens, domínios e paths que as urls benignas.

- **Entropy_Domain:** Normalmente sites maliciosos substituem caracteres na url para fazê-la parecer legítima (por exemplo, 'city' torna-se 'c1ty'). Essa mudança varia o valor da entropia da palavra, com base na pouca variação do alfabeto da língua inglesa.
- **argPathRatio:** O tamanho (contagem de caracteres) do argumento (parte do path após um '?') da url dividido pelo tamanho do path da url.
- **ArgUrlRatio:** O tamanho do argumento da url dividido pelo tamanho da url.

- **argDomanRatio:** O tamanho do argumento da url dividido pelo tamanho do domínio url. A palavra ‘Doman’ aparenta ser um erro de digitação da palavra ‘Domain’.
- **pathurlRatio:** O tamanho do path da url dividido pelo tamanho da url.
- **CharacterContinuityRate:** Determinada pela soma do tamanho do maior token de um certo tipo de caracteres na URL dividida pelo tamanho da URL. Um exemplo seria abc567ti = (3 + 3 + 1)/9 = 0.77 . A razão da continuidade de caracteres é afetada, assim como a entropia do domínio, também pela substituição de caracteres na url.
- **NumberRate_FileName:** Proporção de dígitos na parte ‘Filename’ da url, se houver.
- **domainUrlRatio:** O tamanho do domínio da url dividido pelo tamanho da url.
- **NumberRate_URL:** Proporção de dígitos da url em si.
- **pathDomainRatio:** O tamanho do path da url dividido pelo tamanho do domínio da url.
- **NumberRate_AfterPath:** Proporção de dígitos na parte ‘AfterPath’ da url.
- **avgpathtokenlen:** Média dos tamanhos dos tokens do path da url.

3.2 Vetor de características e classes

Como todos os atributos têm como valores números inteiros ou reais, o vetor de características pode ser composto pelo valor real dos atributos selecionados da seção anterior (*real-valued features*) . Os rótulos das classes são os valores dos campos que identificam os diferentes tipos de urls: ‘benign’, ‘phishing’, ‘spam’ e ‘Defacement’, categorizando o problema como multi-classe.

Querylength	Entropy_Domain	argPathRatio	ArgUrlRatio	argDomanRatio	pathurlRatio	...	NumberRate_URL	pathDomainRatio	NumberRate_AfterPath	avgpathtokenlen	URL_Type_obf_Type
0	0.784493	0.076923	0.034483	0.080000	0.448276	...	0.017241	1.040000	-1.000000	4.400000	Defacement
0	0.784493	0.058824	0.030303	0.080000	0.515151	...	0.000000	1.360000	-1.000000	6.000000	Defacement
0	0.784493	0.060606	0.030769	0.080000	0.507692	...	0.000000	1.320000	-1.000000	5.800000	Defacement
0	0.784493	0.025974	0.018349	0.080000	0.706422	...	0.000000	3.080000	-1.000000	5.500000	Defacement
0	0.784493	0.040816	0.024691	0.080000	0.604938	...	0.000000	1.960000	-1.000000	7.333334	Defacement
...	...	...	...	...	...	...	...	...	...	...	...
29	0.791265	0.752212	0.582192	3.269231	0.773973	...	0.212329	4.346154	0.066667	3.666667	spam
0	0.820010	0.016393	0.013605	0.111111	0.829932	...	0.142857	6.777778	-1.000000	8.461538	spam
58	0.801139	0.838710	0.739837	8.272727	0.882114	...	0.231707	9.863636	0.029412	3.375000	spam
35	0.897617	0.755319	0.612069	4.733333	0.810345	...	0.215517	6.266667	0.418182	3.600000	spam
40	0.801139	0.828283	0.722467	7.454546	0.872247	...	0.229075	9.000000	0.060000	3.250000	spam

[36707 rows x 13 columns]

Figura 2. Colunas representando os elementos do vetor de características, com a última coluna representando a label da classe.

3.3 Distribuição dos dados

São, no total, 36.707 urls com características extraídas no dataset. A distribuição dos dados por classe é representada no gráfico de barras abaixo. Note que a quantidade de exemplos de cada classe é mostrada ao lado de cada barra do gráfico. O script usado para a geração do gráfico se encontra no Apêndice 1.

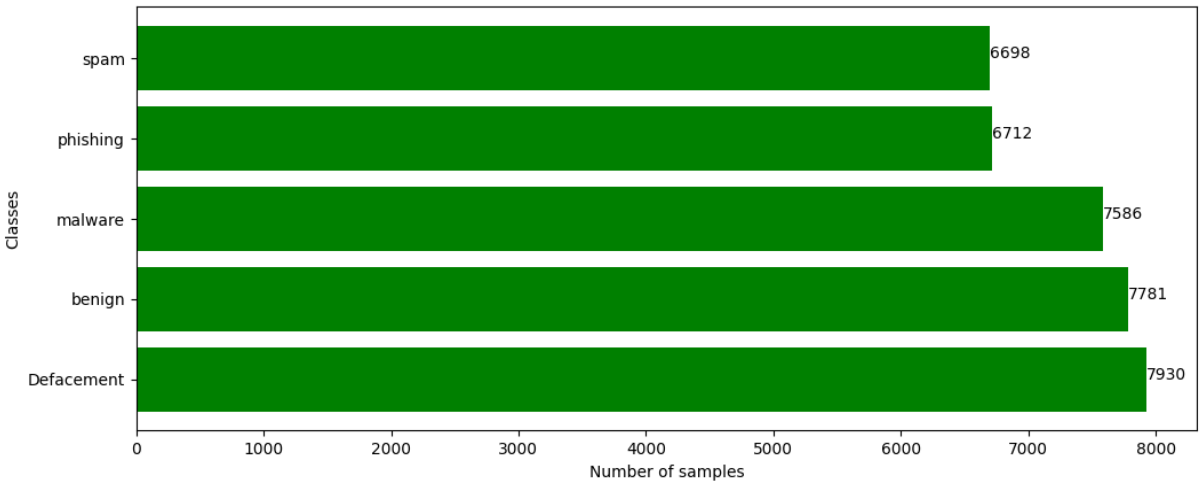

Figura 3. Gráfico de barras da distribuição dos dados do dataset

Referências

[1] Dataset utilizado:https://www.unb.ca/cic/datasets/url-2016.html
[2] Paper “Detecting Malicious URLs Using Lexical Analysis”: https://www.researchgate.net/publication/308365207_Detecting_Malicious_URLs_Using_Lexical_Analysis

Apêndice 1 - Script para geração do gráfico de barras

```
#!/usr/bin/env python
"""
Script responsável para plotar um grafico de barras com a distribuição dos dados de um
dataset
"""
import sys
import pandas
import matplotlib.pyplot as plt


def create_distribution_graph(csv_file, class_column_name):
    """
    Plots a barh graph with the class distribution of a csv file dataset

    :param str csv_file: path to the csv file
    :param str class_column_name: the name of the csv column with the class labels
    """


    # Reads CSV file with data
    csv_data = pandas.read_csv(csv_file)

    #Create class name/label array and class count vectors
    class_names = csv_data[class_column_name].unique()
    class_counts = csv_data[class_column_name].value_counts()

    #Plot bar graph to show class sample distribution
    plt.ylabel('Classes')
    plt.xlabel('Number of samples')
    plt.barh(class_names, class_counts, color='green')

    #Shows the exact class sample count on top of the bar
    for index, value in enumerate(class_counts):
        plt.text(value, index,str(value))

    plt.show()


def do_main(csv_file, class_column_name):

    create_distribution_graph(
        csv_file=csv_file,
        class_column_name=class_column_name,
    )

if __name__ == '__main__':

    if len(sys.argv) < 3:
        exit("Usage: create_distribution_graph.py <csv_file> <class_column_name>")

    do_main(
        csv_file=sys.argv[1],
        class_column_name=sys.argv[2]
```

```
)
```