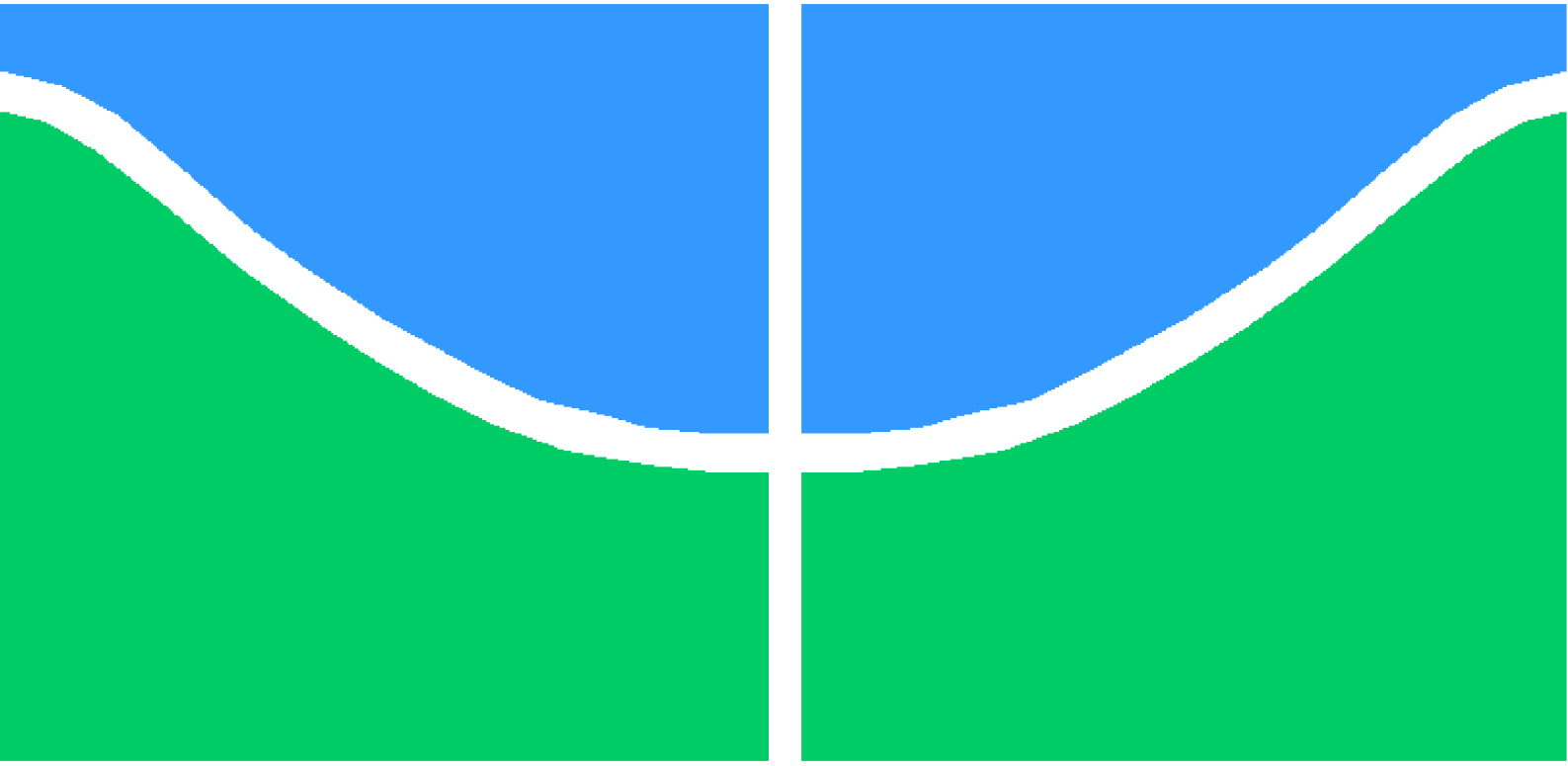

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Uso de redes neurais artificiais para reconhecimento de padrões de malware em fluxos de rede

Autor: Macário Soares da Cruz Júnior

Orientador: Dr. Nilton Correia da Silva

Brasília, DF

2016

Macário Soares da Cruz Júnior

Uso de redes neurais artificiais para reconhecimento de padrões de malware em fluxos de rede

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dr. Nilton Correia da Silva

Brasília, DF

2016

Macário Soares da Cruz Júnior

Uso de redes neurais artificiais para reconhecimento de padrões de malware em fluxos de rede/ Macário Soares da Cruz Júnior. – Brasília, DF, 2016-

68 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Nilton Correia da Silva

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2016.

1. Redes Neurais Artificiais. 2. Machine Learning. I. Dr. Nilton Correia da Silva. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Uso de redes neurais artificiais para reconhecimento de padrões de malware em fluxos de rede

CDU 02:141:005.6

Macário Soares da Cruz Júnior

Uso de redes neurais artificiais para reconhecimento de padrões de malware em fluxos de rede

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 01 de junho de 2016:

Dr. Nilton Correia da Silva
Orientador

Dr. Fabricio Ataides Braz
Convidado 1

Dr. Luis Augusto Laranjeira
Convidado 2

Brasília, DF
2016

Este trabalho é dedicado a Deus, à minha mãe que não poderá estar ao meu lado neste momento e ao meu pai que me deu todo o suporte para que eu chegasse neste estágio da vida.

Agradecimentos

Agradeço primeiramente a Deus por ter me possibilitado chegar a um trabalho de conclusão de curso e por sempre se fazer presente em todas as etapas da minha vida.
Seguntamente a todas as pessoas que de alguma forma contribuíram para este trabalho.
Agradeço à minha família que sempre me apoiou e me deu suporte nos meus momentos mais difíceis, especialmente meu pai Macário por ter me passado forças nos momentos difíceis e por ter me dado o sonho de ser engenheiro. A todos os meus amigos de longa data por se fazerem presentes sempre em minhas necessidades e angústias.
Agradeço a minha namorada Rebecca por estar sempre ao meu lado e me dar forças para nunca desistir. Agradeço também, aos meus sogros Ubirajara e Susan que sempre me trataram como um filho, me aconcelhando e apoiando em minhas decisões.
Agradeço por último a todos os meus amigos de graduação, em especial os que me apoiam cotidianamente: Matheus, Luiz, Parley, Anna Larissa, Lucas e especialmente Gustavo por ter me cobrado e incentivado para a finalização deste trabalho.

Resumo

Este trabalho vem apresentar uma nova proposta de detecção de padrões de malwares em fluxos de rede baseada em redes neurais artificiais, porém utilizando-se de conceitos de aprendizado de maquina , tais como por exemplo *feature engineering* na organização dos modelos de dados a serem utilizados para o treinamento da rede. O objetivo é ao fim do projeto, possuir um sistema inteligente capaz de detectar os padrões de malware presentes em pacotes de fluxos de rede e se possível classifica-los, para que o administrador de rede possua um melhor feedback que uma simples marcação de chance de um pacote ser malicioso.

Palavras-chaves: rede neural artificial. aprendizado de máquina. malware. feature engineering. fluxos de rede.

Abstract

This paper presents a new proposal for detection of malware patterns in network flows based on artificial neural networks, but using machine learning concepts such as feature engineering on the organization of data models to be used for training the network. The goal is once finished the project, have an intelligent system capable of detecting malware patterns present in packets of network flows and can classify them, so that the network administrator can have a better feedback than a simple dial chance of a packet to be malicious.

Key-words: artificial neural network. machine learning. malware. feature engineering. network flows.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

IDS, *Intrusion Detection Systems*

ML, *Machine Learning*

RNA, Redes Neurais Artificiais

MLP, *Multi Layer Perceptron*

Sumário

Parte I

Introdução

1 Introdução

Com o grande crescimento da conexão de computadores ao redor do mundo, é verificado um aumento significativo nos tipos e números de ataques a sistemas conectados em rede, o que gera uma complexidade maior para se planejar mecanismos de prevenção tradicionais. Baseado nesta crescente foram também bastante difundidos os sistemas de detecção de intrusão, que passaram a se tornar um componente importante de diversos sistemas de segurança.

Este trabalho tem a proposta de através do uso de redes neurais artificiais aplicar um modelo de detecção que aprende a detectar tanto padrões conhecidos quanto novos padrões de ataques efetuados.

Nas proximas sessões serão descritos os objetivos do trabalho e em seguida a fundamentação teórica e metodologia abordadas.

Parte II

Objetivos

2 Objetivos

2.1 Objetivo Geral

Este trabalho tem como objetivo apresentar um modelo de *feature engineering* para normalização dos dados capaz de treinar uma rede neural artificial, focanda em detecção de padrões de malwares em fluxos de rede.

2.2 Objetivos específicos

- Definir *features* a partir de dados brutos colhidos num banco de dados;
- Definir um modelo de dados de treinamento a partir dos resultados da *feature engineering*;
- Treinar uma rede neural artificial utilizando os modelo de dados normalizado;
- Testar a rede neural artificial com objetivo de reconhecer padrões de malware.

Parte III

Fundamentação Teórica

3 Fundamentação Teórica

3.1 Intrusion Detection Systems *IDS*

Sistemas de detecção de intrusão, ou Intrusion Detection Systems (*IDS*), são sistemas capazes de detectar atividades suspeitas em pacotes de dados a partir da análise de regras de uma base de conhecimento. Existem dois tipos bastante difundidos de formas de um *IDS* se comportar para a detecção de malwares, o primeiro baseado em detecção por assinatura e o segundo baseado em detecção por anomalias.

- O *IDS* baseado em detecção por assinatura funcionam da seguinte maneira: o software busca em sua base de dados de assinaturas suspeitas, pelas assinaturas recebidas nos pacotes de dados, caso ele encontre, o pacote é marcado como malicioso. (REHMAN, 2003)

- O *IDS* baseado em detecção por anomalia funciona da seguinte maneira: a partir da leitura de muitos fluxos de pacotes, o sistema obtem um padrão de comportamento das atividades, no momento em que um pacote se comporta de maneira diferente o *IDS* capta essa diferença e a trata como anomalia, marcando o pacote. (DEBAR, 2000)

Entretanto este tipo de sistema baseado em assinaturas tem o problema de que a cada vez que um novo tipo de ataque é feito é necessária a atualização das regras de comparação, isso gera um grande esforço para que sejam atualizadas essa base de regras e a base de dados, o que pode se tornar mais crítico com a demora para estas atualizações. Outro problema associado a *IDS*'s é que normalmente eles somente emitem sinais de que um pacode pode conter uma atividade maliciosa, e não chega a categorizá-la.

Os *IDS* são categorizados de acordo com a fonte de dados principal da qual consomem. Sendo assim, as duas categorias mais difundidas são:

- *Network IDS* - Também conhecidos como NIDS, estes sistemas de detecção são ligados na rede que se deseja monitorar, seja ela cabeada ou wifi. Estes detectores capturam pacotes de dados que transitam pela rede para sua análise e a partir daí pode assumir, normalmente o comportamento de detecção por assinatura.

- *Host IDS* - Também conhecidos como HIDS, estes sistemas de detecção estão ligados diretamente a um servidor, podendo fazer suas análises somente nesse servidor sem poder fazer acesso a subrede de dados. Estes detectores, normalmente se comportam

detectando intrusões por anomalias encontradas nos fluxos de dados de arquivos e logs do servidor. (REHMAN, 2003)

3.2 Inteligência artificial

3.2.1 O que é?

A inteligência artificial pode ser definida sob diversos pontos de vista os quais podem dizer respeito tanto à capacidade de pensamento quanto à capacidade raciocínio dos agentes inteligentes quando comparados a seres humanos no caso de pensamentos, e a sistemas ideais no caso do raciocínio. (NORVIG; RUSSELL, 2004) Sendo assim, pode-se entender inteligência artificial como o ramo da computação onde se propõe criar dispositivos inteligentes capazes de simular uma atividade humana, sendo ela um pensameno, um raciocínio ou mesmo uma atitude.

3.2.2 Redes Neurais Artificiais

3.2.2.1 Inspiração biológica

O cérebro humano possui um tipo específico de célula que aparentemente não se regenera lentamente como as outras. A estas células são atribuidas nossa capacidade de pensamento, lembranças e transferência de todo tipo de informações para todo o corpo. Estas células chamadas neurônios estão presentes em cifras de 100 bilhões de unidades. (ANDERSON; MCNEILL, 1992) O neurônio é dividido em basicamente três partes: núcleo, axônio e dendritos, e cada uma delas tem sua atividade bem estipulada. O núcleo é onde ocorre todo o processamento dos sinais elétricos recebidos pelos dendritos que ficam nas extremidades da célula e são transmitidos pelo axônio para o próximo neurônio.

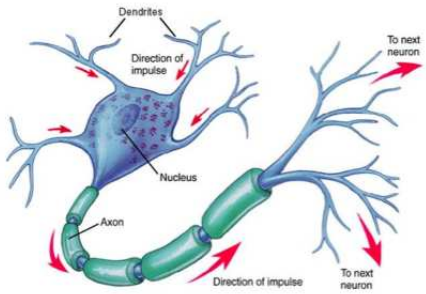

Figura 1: Representação de neurônio humano

Fonte: <http://goo.gl/vbd7T6>

Estes pulsos, ou sinais elétricos, são transmitidos do axônio de um neurônio para os dendritos de um outro neurônio através das fendas sinápticas, ou sinápses.

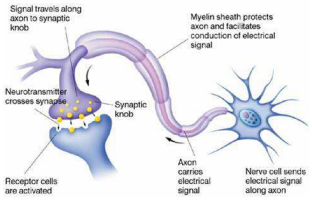

Figura 2: Representação de uma sinápse.

Fonte : <http://goo.gl/TCtvVG>

As sinápses são regiões ativas eletroquimicamente entre as membranas celulares dos neurônios, onde a partir de uma excitação ocorre uma reação química que transmite os sinais de uma célula a outra por intermédio de substâncias chamadas neurotransmissores.

3.2.2.2 Modelagem matemática

O neurônio biológico pode ser modelado matematicamente de um modo que inspire a criação de um neurônio artificial a partir dessa divisão. Sendo assim, uma modelagem matemática de um neurônio biológico é: (ROCHA, 2006)

- Entrada: São os dendritos, por onde os sinais chegam;
- Pesos: São as áreas onde as informações são transferidas de um neurônio para outro, ou seja as sinápses;
- Soma: É o núcleo do neurônio, onde cada entrada é multiplicada com seu devido peso para que em seguida passe por uma função de transferência, a qual vai gerar os sinais de saída dos axônios;
- Função de Transferência: É o potencial necessário para ativação das fendas sinápticas.
- Saída: São os axônios do neurônio biológico.

Dessa maneira, tomando por base este modelo, o desenho de um neurônio artificial seria:

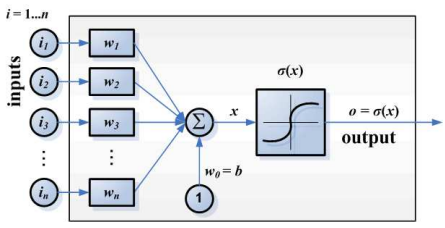

Figura 3: Representação matemática do neurônio.

Fonte : <http://goo.gl/iD2fg4>

3.2.2.3 Perceptron de Multiplas Camadas

O *perceptron* é a menor unidade de processamento, e representa um neurônio conforme a representação da figura 3(3.2.2.2). Uma rede perceptron de múltiplas camadas, ou MLP, é composta de camadas de *perceptrons* alinhados em diferentes camadas, que podem variar a partir de três. A primeira é a camada de entrada, por onde os dados entram na rede, a segunda é uma camada intermediária de processamento, normalmente chamada de camada escondida, e a ultima é a camada dos neurônios de saída, que mostram a resposta. Um modelo básico de MLP é descrito na figura 4(3.2.2.3) a seguir:

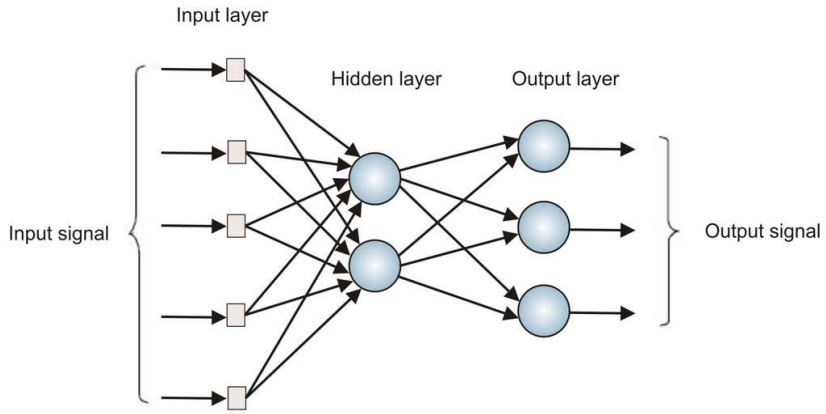

Figura 4: Representação de uma MLP.

Fonte : <http://goo.gl/ICaZBQ>

3.2.3 Treinamento Backpropagation

Backpropagation é um algorítmo de aprendizagem normalmente aplicado a MLP's que visa a aprendizagem baseada em correção de erros. Isso ocorre devido à retropropagação dos erros de saída de uma RNA pelas camadas anteriores, para que assim sejam balanceados os pesos de entrada da rede. Este algorítmo possui duas fases distintas:

- Fase 1: Propagação dos erros
  Os erros são propagados juntamente com os valores de entrada por todos os neurônios da rede a partir da entrada de dados até a camada de saída.

- Fase 2: Retropropagação dos erros
  Os erros que são encontrados nas camadas de saída, são propagados de volta para as camadas anteriores rebalanceando os pesos de entrada nas camadas superiores da rede. (NETTO, 2006) Dessa forma a rede poderá chegar a resultados mais exatos nas próximas interações.

As duas fases descritas podem ser observadas na figura a seguir:

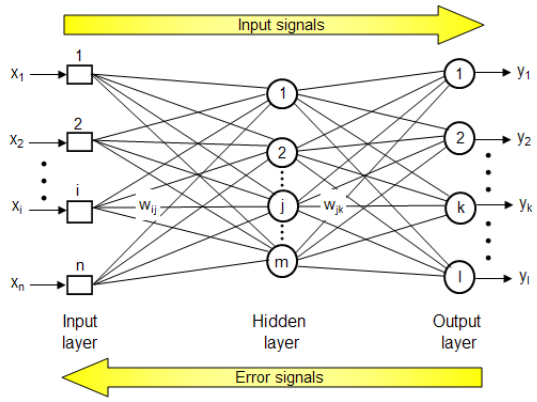

Figura 5: Propagação de sinais de entrada e erros no *Backpropagation*.

Fonte : <https://goo.gl/o7IGy4>

3.3 Machine Learning

Machine learning (ML) é uma sub área da inteligência artificial voltada à otimizar critérios de desempenho de acordo com análise de dados e ocorrências passadas. (ALPAYDIN, 2010) Uma das características mais marcantes da ML é a análise de *datasets* para automatizar o desenvolvimento de modelos analíticos para suas funcionalidades, isso quer dizer que de acordo com essas análises e com dados de ocorrências já conhecidas por uma aplicação baseada em ML, o aprendizado possibilita às aplicações reagirem de maneira autônoma à eventualidades para as quais não foram programadas.

Machine Learning normalmente é utilizada em duas situações que podem ser enxergadas pela análise do problema:

- Complexidade elevada do problema:
  Por exemplo, tarefas rotineiras que seres humanos executam, mas que para serem programadas o algorítmo teria uma complexitade enorme como dirigir ou reconhecer imagens. Outro exemplo é a necessidade de um processamento de uma massa de dados muito grande.

- Necessidade de adaptabilidade do sistema:
  Por exemplo, detecção de diversos tipos de spam, para marcar mensagens. (SHALEV-SHWARTZ; BEN-DAVID, 2014)

3.3.1 *Workflow*

O *workflow* básico de ML consiste em duas fases, a de construção de um modelo e a de predição. Na primeira são utilizados dados históricos (ou dados de treinamento) para um ciclo de modelagem onde será definido, evoluido e otimizado um modelo de dados para que o algorítmo o consuma, realizando assim um tipo de aprendizado. É a partir

deste modelo otmizado que o algorítmo vai conseguir fazer predições sobre novos dados ou ainda categorizações.

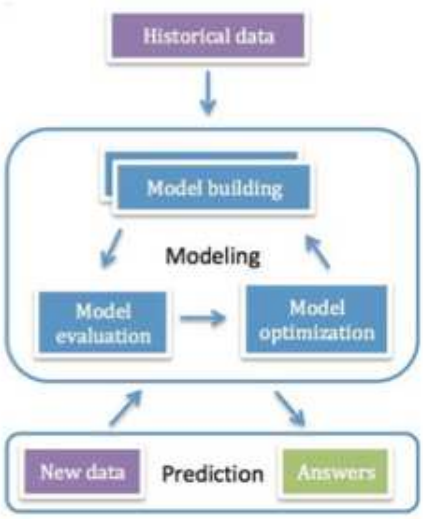

Figura 6: *Workflow* básico de ML.

Fonte : (BRINK; RICHARDS; FETHEROLF, 2015)

3.3.2 Tipos de aprendizado

- Supervisionado: No aprendizado supervisionado, uma massa de dados de treinamento é rodada pelo programa possuindo labels características que as classificam, e no momento em que um novo dado aparece sem essa label (dado de teste), espera-se que o programa seja capaz de predizer qual a classificação do dado.

- Não Supervisionado: No aprendizado não supervisionado, a massa de dados de treinamento é a mesma dos dados de teste, pois estes dados não possuem labels que distingüem características. Dessa forma o programa reconhece as características a cada novo dado entregue e começa a fazer a separação de forma autônoma. (CHAO, 2011)

- Reforço: No aprendizado por reforço, o programa ao receber um sinal de entrada dispara uma ação que muda o valor deste sinal. Assim que o valor do sinal de entrada é alterado e devolvido, mudando assim o estado do ambiente de aprendizagem. A partir disso é recebida uma nova entrada para disparar outra ação que novamente devolverá um valor diferente alterando o estado do ambiente, com a intenç!ao de sempre se aumentar os valores de interação. (KAELBLING; LITTMAN; MOORE, 1996)

Parte IV

Metodologia

4 Metodologia

Tendo como meta a definição de um modelo de ML para ser aplicada em uma RNA a ser treinada para detecção de intrusão, a estratégia a ser seguida é o conceito de *feature engineering* presente dentro de *machine learning*.

4.1 Feature Engineering

No processo básico de *feature engineering* exitem alguns passos a serem tomados.

- Pré-processamento dos dados:
  Neste primeiro passo, é montado o modelo de dados de treinamento que será usado para apresentação e utilização dos dados. Para isso é necessário encontrar uma corelação entre os dados de entrada e o dado a ser atingido pelo processo de aprendizado. A partir do momento que a corelação é encontrada, os dados devem ser dispostos de modo que facilite sua visualização e uso, sendo assim de acordo com (BRINK; RICHARDS; FETHEROLF, 2015), é recomendado que se convertam os dados categóricos (normamente textuais) em dados numéricos, para facilitar seu uso.

- Treinamento do modelo:
  O modelo proposto deve passar por um breve treinamento utilizando-se de algorítmos de ML que verificam a aplicabilidade do modelo de dados para se construir um modelo de ML. Estes algorítmos utilizam os chamados *tuning parameters*, que são os parâmetros utilizados pelo algorítmo durante a leitura dos dados e são utilizados para definir o quão complexa é a relação entre os dados de entrada e o valor alvo. (BRINK; RICHARDS; FETHEROLF, 2015)

- Avaliar o modelo:
  Uma vez com o modelo pronto é necessário avaliar a aplicabilidade do modelo em vista dos dados que se querem atingir. Um exemplo simples de avaliação é a comparação dos dados que deveriam ter sido atingidos com os dados preditos pelo modelo, como exemplifica a tabela a seguir:

Tabela 1: Class-wise accuracy

Test set Labels	Value equals	Predictions
1	True	1
0	True	0
1	True	1
1	False	0

Esta tabela simplificada mostra uma forma fácil e rápida de calcular se o modelo está gerando os resultados esperados. Neste caso, com 3 acertos em 4 valores tem-se 75% de acerto. (BRINK; RICHARDS; FETHEROLF, 2015)

- Escolher as *features*:
  Neste momento as *features* que deverão ser utilizadas dentro do modelo de ML são selecionadas.

4.1.1 Como escolher as features

Existem várias formas diferentes de se escolher um bom conjunto de *features*, aqui serão listados os dois algorítmos mais difundidos:

- Algorítmo *Forwards Selection*:
  Consiste em aplicar um algorítmo no conjunto total de *features* que vai testar uma por uma selecionando para um novo grupo de *features* que serão utilizadas somente as que possuem maior eficácia na avaliação de predição de valores.

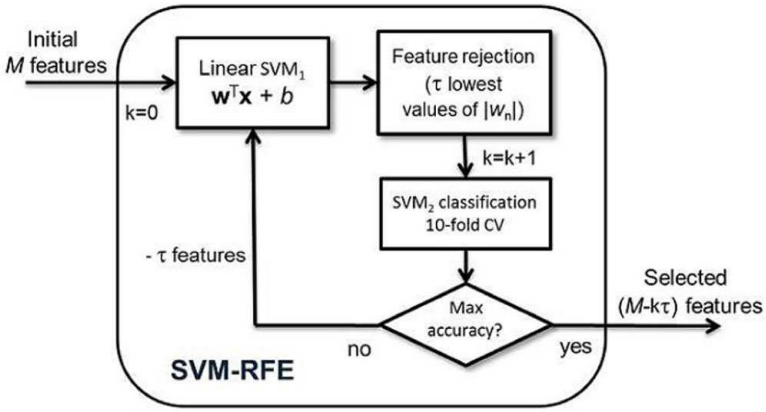

Figura 7: Representação do ciclo de *Forward-selection*.

Fonte : <http://goo.gl/270zF3>

- Algorítmo *Backwards Elimination*:
  Consiste em aplicar um algorítmo no conjunto total de *features* que vai testar uma por uma selecionando para um novo grupo de *features* que serão eliminadas somente as que possuem menor eficácia na avaliação de predição de valores.

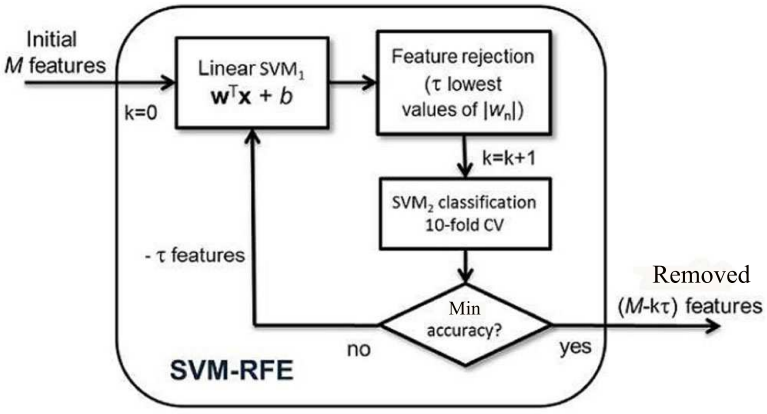

Figura 8: Representação do ciclo de *Forward-selection*.

Fonte : <http://goo.gl/270zF3>

Segundo (BRINK; RICHARDS; FETHEROLF, 2015), estes algorítmos não garantem o melhor conjunto de features possível, porém eles entregam um conjunto aproximado das melhores features encontradas em cada iteração.

Parte V

Resultados

5 Resultados

De acordo com a proposta de *feature engineering* apresentada e utilizando uma amostra de dados de fluxos de rede este capítulo apresentará os resultados atingidos até esta etapa do trabalho.

5.1 Features

Como no modelo de dados utilizado para definição das *features* é uma lista de dados retornada por uma consulta em um banco de dados, as *features* definidas para representar um fluxo de dados, que poderá ser definido como malicioso ou não, serão expostas com a mesma nomenclatura do retorno da consulta:

```
column=flow:avg_packet_size;

column=flow:bytes;

column=flow:detected_protocol;

column=flow:packets;

column=event:classification_id;

column=event:priority_id;

column=event:signature_id.
```

Estas *features* foram escolhidas pois elas definem um fluxo comum, no caso da leitura e aprendizagem de column=flow , e sempre que no mesmo registro a coluna column=event aparecer, para o processo de aprendizado significará que este é um tipo de fluxo que deverá ser marcado como malicioso, e é basicamente definido pelas *features* escolhidas.

Os próximos passos seguindo o conceito de *feature engineering* seriam o treinamento e avaliação deste modelo, que neste momento ainda não podem ser completos pois esta definição, durante o desenvolvimento deste trabalho, não é somente para ser aplicada nos conceitos de ML, mas para fazer a separação dos dados e treinar uma RNA, que ainda será produzida.

Referências

ALPAYDIN, E. *Introduction to Machine Learning, 2nd edn. Adaptive Computation and Machine Learning*. [S.l.]: The MIT Press (February 2010), 2010. Citado na página 35.

ANDERSON, D.; MCNEILL, G. Artificial neural networks technology. *Kaman Sciences Corporation*, v. 258, n. 6, 1992. Citado na página 32.

BRINK, H.; RICHARDS, J. W.; FETHEROLF, M. *Real-world machine learning*. [S.l.: s.n.], 2015. Citado 4 vezes nas páginas 36, 39, 40 e 41.

CHAO, W.-L. Machine learning tutorial. National Taiwan University, 2011. Citado na página 36.

DEBAR, H. An introduction to intrusion-detection systems. *Proceedings of Connect*, v. 2000, 2000. Citado na página 31.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. *Journal of artificial intelligence research*, v. 4, p. 237–285, 1996. Citado na página 36.

NETTO, R. S. Detecção de intrusão utilizando redes neurais artificiais no reconhecimento de padrões de ataque. Universidade de Itajuba, 2006. Citado na página 34.

NORVIG, P.; RUSSELL, S. *Inteligência Artificial, 2ª Edição*. [S.l.]: Elsevier Brasil, 2004. Citado na página 32.

REHMAN, R. U. *Intrusion detection systems with Snort: advanced IDS techniques using Snort, Apache, MySQL, PHP, and ACID*. [S.l.]: Prentice Hall Professional, 2003. Citado na página 31.

ROCHA, D. L. R. Utilizacao de um ambiente de honeynet no treinamento de redes neurais artificiais para deteccao de intrusao. Universidade de Brasilia, 2006. Citado na página 33.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. *Understanding machine learning: From theory to algorithms*. [S.l.]: Cambridge University Press, 2014. Citado na página 35.

Apêndices

APÊNDICE A – Amostra de Dados

Listing A.1: Amostra de dados, cedidos pelo laboratório LADES da Universidade de Brasília no campus Gama, que representam 4 registros de banco de dados.

```
1454451544027.2547497053
 column=flow:avg_inter_time, timestamp=1463150618900, value=6697

 1454451544027.2547497053
 column=flow:avg_packet_size, timestamp=1463150618900, value=60

 1454451544027.2547497053                                column=flow:
    bytes,
timestamp=1463150618900, value=540

 1454451544027.2547497053                                column=flow:
    detected_os,
timestamp=1463150618900, value=

 1454451544027.2547497053
 column=flow:detected_protocol, timestamp=1463150618900, value
    =7/HTTP

 1454451544027.2547497053
 column=flow:detection_completed, timestamp=1463150618900, value
    =0

 1454451544027.2547497053                                column=flow:
    first_seen,
timestamp=1463150618900, value=1454451544027

 1454451544027.2547497053                                column=flow:
    flow_duration,
timestamp=1463150618900, value=60283

 1454451544027.2547497053
 column=flow:host_server_name, timestamp=1463150618900, value=
```

```
 1454451544027.2547497053                column=flow:
    http_method,
timestamp=1463150618900, value=0

 1454451544027.2547497053                column=flow:
    inter_time-0,
timestamp=1463150618900, value=0

 1454451544027.2547497053                column=flow:
    inter_time-1,
timestamp=1463150618900, value=233

 1454451544027.2547497053                column=flow:
    inter_time-2,
timestamp=1463150618900, value=472

 1454451544027.2547497053                column=flow:
    inter_time-3,
timestamp=1463150618900, value=944

 1454451544027.2547497053                column=flow:
    inter_time-4,
timestamp=1463150618900, value=1892

 1454451544027.2547497053                column=flow:
    inter_time-5,
timestamp=1463150618900, value=3780

 1454451544027.2547497053                column=flow:
    inter_time-6,
timestamp=1463150618900, value=7568

 1454451544027.2547497053                column=flow:
    inter_time-7,
timestamp=1463150618900, value=15121

 1454451544027.2547497053                column=flow:
    inter_time-8,
timestamp=1463150618900, value=30273
```

```
 1454451544027.2547497053                                column=flow:
    last_seen,
timestamp=1463150618900, value=1454451604310

 1454451544027.2547497053                                column=flow:
    lower_ip,
timestamp=1463150618900, value=]\xB8\xD7\x97

 1454451544027.2547497053                                column=flow:
    lower_name,
timestamp=1463150618900, value=93.184.215.151

 1454451544027.2547497053                                column=flow:
    lower_port,
timestamp=1463150618900, value=80

 1454451544027.2547497053
 column=flow:max_packet_size, timestamp=1463150618900, value=60

 1454451544027.2547497053
 column=flow:min_packet_size, timestamp=1463150618900, value=60

 1454451544027.2547497053                                column=flow:
    packet_size-0,
timestamp=1463150618900, value=60

 1454451544027.2547497053                                column=flow:
    packet_size-1,
timestamp=1463150618900, value=60

 1454451544027.2547497053                                column=flow:
    packet_size-2,
timestamp=1463150618900, value=60

 1454451544027.2547497053                                column=flow:
    packet_size-3,
timestamp=1463150618900, value=60
```

```
 1454451544027.2547497053                                    column=flow:
    packet_size-4,
timestamp=1463150618900, value=60

 1454451544027.2547497053                                    column=flow:
    packet_size-5,
timestamp=1463150618900, value=60

 1454451544027.2547497053                                    column=flow:
    packet_size-6,
timestamp=1463150618900, value=60

 1454451544027.2547497053                                    column=flow:
    packet_size-7,
timestamp=1463150618900, value=60

 1454451544027.2547497053                                    column=flow:
    packet_size-8,
timestamp=1463150618900, value=60

 1454451544027.2547497053                                    column=flow:
    packets,
timestamp=1463150618900, value=9

 1454451544027.2547497053
 column=flow:packets_without_payload, timestamp=1463150618900,
    value=0

 1454451544027.2547497053
 column=flow:payload_avg_size, timestamp=1463150618900, value=46

 1454451544027.2547497053                                    column=flow:
    payload_bytes,
timestamp=1463150618900, value=414

 1454451544027.2547497053
 column=flow:payload_first_size, timestamp=1463150618900, value
    =46
```

```
 1454451544027.2547497053
 column=flow:payload_max_size, timestamp=1463150618900, value=46

 1454451544027.2547497053
 column=flow:payload_min_size, timestamp=1463150618900, value=46

 1454451544027.2547497053                                column=flow:
    protocol,
timestamp=1463150618900, value=TCP

 1454451544027.2547497053                                column=flow:
    upper_ip,
timestamp=1463150618900, value=\xAC\x10\x05\xC7

 1454451544027.2547497053                                column=flow:
    upper_name,
timestamp=1463150618900, value=172.16.5.199

 1454451544027.2547497053                                column=flow:
    upper_port,
timestamp=1463150618900, value=62199

 1454451544027.2547497053                                column=flow:
    vlan_id,
timestamp=1463150618900, value=0

 1454494277630.17109164
 column=flow:avg_inter_time, timestamp=1463150620774, value=0

 1454494277630.17109164
 column=flow:avg_packet_size, timestamp=1463150620774, value=76

 1454494277630.17109164                                  column=flow:
    bytes,
timestamp=1463150620774, value=76

 1454494277630.17109164                                  column=flow:
    detected_os,
timestamp=1463150620774, value=
```

```
1454494277630.17109164
column=flow:detected_protocol, timestamp=1463150620774, value
    =5/DNS

1454494277630.17109164
column=flow:detection_completed, timestamp=1463150620774, value
    =1

1454494277630.17109164
column=flow:dns_bad_packet, timestamp=1463150620774, value=0

1454494277630.17109164
column=flow:dns_num_answers, timestamp=1463150620774, value=0

1454494277630.17109164
column=flow:dns_num_queries, timestamp=1463150620774, value=1

1454494277630.17109164
column=flow:dns_query_class, timestamp=1463150620774, value=1

1454494277630.17109164
column=flow:dns_query_type, timestamp=1463150620774, value=1

1454494277630.17109164                                  column=flow:
    dns_ret_code,
timestamp=1463150620774, value=0

1454494277630.17109164                                  column=flow:
    dns_rsp_type,
timestamp=1463150620774, value=0

1454494277630.17109164                                  column=flow:
    first_seen,
timestamp=1463150620774, value=1454494277630

1454494277630.17109164                                  column=flow:
    flow_duration,
timestamp=1463150620774, value=0
```

```
 1454494277630.17109164
 column=flow:host_server_name, timestamp=1463150620774,
value=wpad.localdomain

 1454494277630.17109164                                    column=flow:
    inter_time-0,
timestamp=1463150620774, value=0

 1454494277630.17109164                                    column=flow:
    last_seen,
timestamp=1463150620774, value=1454494277630

 1454494277630.17109164                                    column=flow:
    lower_ip,
timestamp=1463150620774, value=\xAC\x10\x05\x01

 1454494277630.17109164                                    column=flow:
    lower_name,
timestamp=1463150620774, value=172.16.5.1

 1454494277630.17109164                                    column=flow:
    lower_port,
timestamp=1463150620774, value=53

 1454494277630.17109164
 column=flow:max_packet_size, timestamp=1463150620774, value=76

 1454494277630.17109164
 column=flow:min_packet_size, timestamp=1463150620774, value=76

 1454494277630.17109164                                    column=flow:
    packet_size-0,
timestamp=1463150620774, value=76

 1454494277630.17109164                                    column=flow:
    packets,
timestamp=1463150620774, value=1
```

```
 1454494277630.17109164
 column=flow:packets_without_payload, timestamp=1463150620774,
    value=0

 1454494277630.17109164
 column=flow:payload_avg_size, timestamp=1463150620774, value=62

 1454494277630.17109164                                  column=flow:
    payload_bytes,
timestamp=1463150620774, value=62

 1454494277630.17109164
 column=flow:payload_first_size, timestamp=1463150620774, value
    =62

 1454494277630.17109164
 column=flow:payload_max_size, timestamp=1463150620774, value=62

 1454494277630.17109164
 column=flow:payload_min_size, timestamp=1463150620774, value=62

 1454494277630.17109164                                  column=flow:
    protocol,
timestamp=1463150620774, value=UDP

 1454494277630.17109164                                  column=flow:
    upper_ip,
timestamp=1463150620774, value=\xAC\x10\x05Q

 1454494277630.17109164                                  column=flow:
    upper_name,
timestamp=1463150620774, value=172.16.5.81

 1454494277630.17109164                                  column=flow:
    upper_port,
timestamp=1463150620774, value=55896

 1454494277630.17109164                                  column=flow:
    vlan_id,
```

```
timestamp=1463150620774, value=0

 1454498806893.17109164
 column=flow:avg_inter_time, timestamp=1463150617335, value=0

 1454498806893.17109164
 column=flow:avg_packet_size, timestamp=1463150617335, value=80

 1454498806893.17109164                                column=flow:
    bytes,
timestamp=1463150617335, value=80

 1454498806893.17109164                                column=flow:
    detected_os,
timestamp=1463150617335, value=

 1454498806893.17109164
 column=flow:detected_protocol, timestamp=1463150617335,
value=5.126/DNS.Google

 1454498806893.17109164
 column=flow:detection_completed, timestamp=1463150617335, value
    =1

 1454498806893.17109164                                column=flow:
    first_seen,
timestamp=1463150617335, value=1454498806893

 1454498806893.17109164                                column=flow:
    flow_duration,
timestamp=1463150617335, value=0

 1454498806893.17109164
 column=flow:host_server_name, timestamp=1463150617335, value=
clients.l.google.com

 1454498806893.17109164                                column=flow:
    inter_time-0,
timestamp=1463150617335, value=0
```

```
 1454498806893.17109164                                    column=flow:
    last_seen,
timestamp=1463150617335, value=1454498806893

 1454498806893.17109164                                    column=flow:
    lower_ip,
timestamp=1463150617335, value=\xAC\x10\x05\x01

 1454498806893.17109164                                    column=flow:
    lower_name,
timestamp=1463150617335, value=172.16.5.1

 1454498806893.17109164                                    column=flow:
    lower_port,
timestamp=1463150617335, value=53

 1454498806893.17109164
 column=flow:max_packet_size, timestamp=1463150617335, value=80

 1454498806893.17109164
 column=flow:min_packet_size, timestamp=1463150617335, value=80

 1454498806893.17109164                                    column=flow:
    packet_size-0,
timestamp=1463150617335, value=80

 1454498806893.17109164                                    column=flow:
    packets,
timestamp=1463150617335, value=1

 1454498806893.17109164
 column=flow:packets_without_payload, timestamp=1463150617335,
    value=0

 1454498806893.17109164
 column=flow:payload_avg_size, timestamp=1463150617335, value=66
```

```
 1454498806893.17109164                                    column=flow:
    payload_bytes,
timestamp=1463150617335, value=66

 1454498806893.17109164
 column=flow:payload_first_size, timestamp=1463150617335, value
    =66

 1454498806893.17109164
 column=flow:payload_max_size, timestamp=1463150617335, value=66

 1454498806893.17109164
 column=flow:payload_min_size, timestamp=1463150617335, value=66

 1454498806893.17109164                                    column=flow:
    protocol,
timestamp=1463150617335, value=UDP

 1454498806893.17109164                                    column=flow:
    upper_ip,
timestamp=1463150617335, value=\xAC\x10\x05?

 1454498806893.17109164                                    column=flow:
    upper_name,
timestamp=1463150617335, value=172.16.5.63

 1454498806893.17109164                                    column=flow:
    upper_port,
timestamp=1463150617335, value=56095

 1454498806893.17109164                                    column=flow:
    vlan_id,
timestamp=1463150617335, value=0

 1454498808890.855085000
column=flow:avg_inter_time, timestamp=1463150631358, value=0

 1454498808890.855085000
column=flow:avg_packet_size, timestamp=1463150631358, value=60
```

```
 1454498808890.855085000                                    column=flow:
    bytes,
timestamp=1463150631358, value=60

 1454498808890.855085000                                    column=flow:
    detected_os,
timestamp=1463150631358, value=

 1454498808890.855085000
column=flow:detected_protocol, timestamp=1463150631358, value=7/
   HTTP

 1454498808890.855085000
column=flow:detection_completed, timestamp=1463150631358, value
   =0

 1454498808890.855085000                                    column=flow:
    first_seen,
timestamp=1463150631358, value=1454498808890

 1454498808890.855085000                                    column=flow:
    flow_duration,
timestamp=1463150631358, value=0

 1454498808890.855085000
column=flow:host_server_name, timestamp=1463150631358, value=

 1454498808890.855085000                                    column=flow:
    http_method,
timestamp=1463150631358, value=0

 1454498808890.855085000                                    column=flow:
    inter_time-0,
timestamp=1463150631358, value=0

 1454498808890.855085000                                    column=flow:
    last_seen,
timestamp=1463150631358, value=1454498808890
```

```
 1454498808890.855085000                                    column=flow:
    lower_ip,
timestamp=1463150631358, value=\xC8\x8F\xF72

 1454498808890.855085000                                    column=flow:
    lower_name,
timestamp=1463150631358, value=200.143.247.50

 1454498808890.855085000                                    column=flow:
    lower_port,
timestamp=1463150631358, value=80

 1454498808890.855085000
column=flow:max_packet_size, timestamp=1463150631358, value=60

 1454498808890.855085000
column=flow:min_packet_size, timestamp=1463150631358, value=60

 1454498808890.855085000                                    column=flow:
    packet_size-0,
timestamp=1463150631358, value=60

 1454498808890.855085000                                    column=flow:
    packets,
timestamp=1463150631358, value=1

 1454498808890.855085000
column=flow:packets_without_payload, timestamp=1463150631358,
   value=0

 1454498808890.855085000
column=flow:payload_avg_size, timestamp=1463150631358, value=46

 1454498808890.855085000                                    column=flow:
    payload_bytes,
timestamp=1463150631358, value=46

 1454498808890.855085000
```

```
column=flow:payload_first_size, timestamp=1463150631358, value
    =46

 1454498808890.855085000
column=flow:payload_max_size, timestamp=1463150631358, value=46

 1454498808890.855085000
column=flow:payload_min_size, timestamp=1463150631358, value=46

 1454498808890.855085000                                column=flow:
    protocol,
timestamp=1463150631358, value=TCP

 1454498808890.855085000                                column=flow:
    upper_ip,
timestamp=1463150631358, value=\xAC\x10\x05\xAF

 1454498808890.855085000                                column=flow:
    upper_name,
timestamp=1463150631358, value=172.16.5.175

 1454498808890.855085000                                column=flow:
    upper_port,
timestamp=1463150631358, value=60048

 1454498808890.855085000                                column=flow:
    vlan_id,
timestamp=1463150631358, value=0

 1454498812845.1057296556
 column=event:classification_id, timestamp=1463150617151, value
    =2

 1454498812845.1057296556                               column=event:
    event_id,
timestamp=1463150617151, value=13

 1454498812845.1057296556
```

```
 column=event:event_microsecond, timestamp=1463150617151, value
    =845739

 1454498812845.1057296556                                column=event:
    event_second,
timestamp=1463150617151, value=1454498812

 1454498812845.1057296556                                column=event:
    generator_id,
timestamp=1463150617151, value=119

 1454498812845.1057296556                                column=event:
    priority_id,
timestamp=1463150617151, value=3

 1454498812845.1057296556                                column=event:
    sensor_id,
timestamp=1463150617151, value=0

 1454498812845.1057296556                                column=event:
    signature_id,
timestamp=1463150617151, value=31

 1454498812845.1057296556
 column=flow:avg_inter_time, timestamp=1463150617151, value=0

 1454498812845.1057296556
 column=flow:avg_packet_size, timestamp=1463150617151, value=60

 1454498812845.1057296556                                column=flow:
    bytes,
timestamp=1463150617151, value=120

 1454498812845.1057296556                                column=flow:
    detected_os,
timestamp=1463150617151, value=

 1454498812845.1057296556
```

```
 column=flow:detected_protocol, timestamp=1463150617151, value
    =125/Skype

 1454498812845.1057296556
 column=flow:detection_completed, timestamp=1463150617151, value
    =1

 1454498812845.1057296556                                column=flow:
    first_seen,
timestamp=1463150617151, value=1454498812845

 1454498812845.1057296556                                column=flow:
    flow_duration,
timestamp=1463150617151, value=0

 1454498812845.1057296556
 column=flow:host_server_name, timestamp=1463150617151, value=

 1454498812845.1057296556                                column=flow:
    inter_time-0,
timestamp=1463150617151, value=0

 1454498812845.1057296556                                column=flow:
    inter_time-1,
timestamp=1463150617151, value=0

 1454498812845.1057296556                                column=flow:
    last_seen,
timestamp=1463150617151, value=1454498812845

 1454498812845.1057296556                                column=flow:
    lower_ip,
timestamp=1463150617151, value=\xAC\x10\x05?

 1454498812845.1057296556                                column=flow:
    lower_name,
timestamp=1463150617151, value=172.16.5.63
```

```
 1454498812845.1057296556                                column=flow:
    lower_port,
timestamp=1463150617151, value=51592

 1454498812845.1057296556
 column=flow:max_packet_size, timestamp=1463150617151, value=60

 1454498812845.1057296556
 column=flow:min_packet_size, timestamp=1463150617151, value=60

 1454498812845.1057296556                                column=flow:
    packet_size-0,
timestamp=1463150617151, value=60

 1454498812845.1057296556                                column=flow:
    packet_size-1,
timestamp=1463150617151, value=60

 1454498812845.1057296556                                column=flow:
    packets,
timestamp=1463150617151, value=2

 1454498812845.1057296556
 column=flow:packets_without_payload, timestamp=1463150617151,
    value=0

 1454498812845.1057296556
 column=flow:payload_avg_size, timestamp=1463150617151, value=46

 1454498812845.1057296556                                column=flow:
    payload_bytes,
timestamp=1463150617151, value=92

 1454498812845.1057296556
 column=flow:payload_first_size, timestamp=1463150617151, value
    =46

 1454498812845.1057296556
 column=flow:payload_max_size, timestamp=1463150617151, value=46
```

```
 1454498812845.1057296556
 column=flow:payload_min_size, timestamp=1463150617151, value=46

 1454498812845.1057296556                                column=flow:
    protocol,
timestamp=1463150617151, value=TCP

 1454498812845.1057296556                                column=flow:
    upper_ip,
timestamp=1463150617151, value=\xBF\xECj{

 1454498812845.1057296556                                column=flow:
    upper_name,
timestamp=1463150617151, value=191.236.106.123

 1454498812845.1057296556                                column=flow:
    upper_port,
timestamp=1463150617151, value=80

 1454498812845.1057296556                                column=flow:
    vlan_id,
timestamp=1463150617151, value=0
```