**SOLUÇÕES EM MINERAÇÕES DE DADOS PARA**

**BASE DE DADOS DO UCI MACHINE LEARNING: PHISHING WEBSITES DATA**

Relatório final com objetivo de obtenção de notas da Disciplina Soluções em Mineração de Dados do Curso de Ciências de Dados Analytics – Poli - UPE

Prof Dr Leandro Maciel Almeida

ALUNOS DO GRUPO 3:

* DIOGO DIAS
* ANA BEATRIZ DE ALMEIDA
* ISABELLA PAZ
* LAÍS VIDOTO
* HELMER PAIVA
* MARIA REGINA MENEZES

**RECIFE – PE**

**Julho de 2019**

**SUMÁRIO**

[**1. I**](#_3fxrbqfh7jey)**ntrodução**

[**2.**](#_jnkbz08y3p7g) **Etapas do KDD – Scripts em Python**

**2.1 Seleção**

**2.2. Pré-processamento e Limpeza**

**2.3. Transformação dos dados**

**2.4. Data Mining**

**2.4.1. Arvore de Decisão Simples**

**2.4.2. Random Forest**

**2.4.3. Rede Neural MLP**

**2.4.4. Comitê de Redes Neurais em Python**

**2.5. Interpretação e Avaliação**

**2.5.1. Gráficos de caixa (Boxplot)**

**2.5.2. Analise dos Resultados**

**3. Aspectos Relevantes**

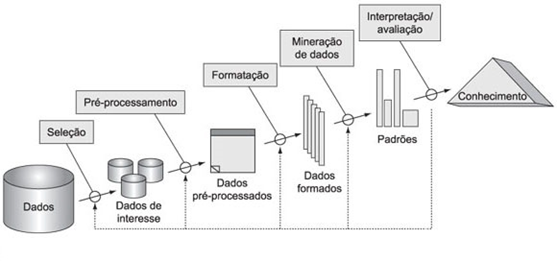
[**Referências**](#_56h4futan5g4)

**1** [**I**](#_3fxrbqfh7jey)**IIGBIIIIntrodução**

A internet é item indispensável na vida contemporânea. Porém, ao se conectarem, os usuários estão vulneráveis a diferentes tipos de ameaças da Web que podem causar tanto danos financeiros, como sociais e emocionais ao rouba-lhes dados privados. Phishing é considerado uma forma de web-ameaças em que um endereço de Web apresenta-se como um site honesto com o objetivo de obter dados confidenciais de valor.

Por conta da alta gama de possibilidades de ataques de phishing e devido os recursos que determinam o tipo de páginas perigosas da Web mudarem constantemente através das atualizações e melhoramentos nos sites maldosos, ainda não há uma solução única que consiga prevenir todas as ameaças e esse tende a ser um problema contínuo.

O presente trabalho pretende portanto gerar um modelo inteligente capaz de prever ataques phishing para a base “Phishing Websites Data Set” do UCI Machine Learning, baseando-se nas etapas do KDD, para testar quatro métodos de mineração de dados (Arvore de Decisão Simples, Random Forest, Rede Neural MLP e Comitê de Redes Neurais em Python) para encontrar dentre eles o modelo que soluciona mais satisfatoriamente o problema.



A base de dados apresenta os recursos mais importantes e que se mostraram sólidos e eficazes na previsão de sites de phishing. Além disso, foi proposto alguns novos recursos, atribuímos experimentalmente novas regras a alguns recursos conhecidos e atualizamos alguns outros recursos.

Quando um site é considerado **SUSPEITO**, isso significa que ele pode ser ou falso ou legítimo, o que significa que o site possui alguns recursos legítimos e fraudulentos.

* Analise realizada na base de dados, escolhida abaixo:

https://archive.ics.uci.edu/ml/datasets/phishing+websites#

* arquivo de utilização-> traning dataset.arf
* Resultados encontrados para cada atributo:
  + 1 – Sim
  + 0 – Talvez/Desconhecido/Não informado
  + -1 – Não
* Numero total de atributos: 31
* y é o campo: Result

**Atributos da base:**

|  |  |
| --- | --- |
| **Nome do atributo** | **Descrição** |
| having\_IP\_Address | Ter endereço de IP |
| URL\_Length | Comprimento do URL |
| Shortining\_Service | Serviço de Shortining |
| having\_At\_Symbol | URL com o símbolo@ |
| double\_slash\_redirecting | Redirecionamento de barra dupla |
| Prefix\_Suffix | Prefixo sufixo |
| having\_Sub\_Domain | Ter subdomínio |
| SSLfinal\_State | Estado SSLfinal |
| Domain\_registeration\_length | Comprimento de registro de domínio – expiração do domínio |
| Favicon | a imagem gráfica (ícone) associada a uma página da Web específica |
| port | Portas abertas |
| HTTPS\_token | Token HTTPS na parte do domínio do URL |
| Request\_URL | URL de solicitação, a página contém objetos externos do tipo imagens. |
| URL\_of\_Anchor | as tags <a> e o site possuem nomes de domínio diferentes. |
| Links\_in\_tags | as tags <Meta>, <Script> e <Link> estão vinculadas a mesma página da Web. |
| SFH | o Server Form Handler (SFH) é uma string vazia ou em branco. |
| Submitting\_to\_email | utiliza as funções ‘mail” ou “mailto” no código-fonte da página da Web para enviar as informações do usuário para o e-mail pessoal do phisher. |
| Abnormal\_URL | o nome do host foi incluído no URL do site examinado. |
| Redirect | quantas vezes um site foi redirecionado |
| on\_mouseover | mostra para o usuário um URL falso na barra de status. |
| RightClick | a função de clicar com o botão direito é desativada para impedir que os usuários visualizem e salvem o código-fonte da página da Web. |
| popUpWidnow | o site solicita que o usuário envie suas informações pessoais através de uma janela pop-up. |
| Iframe | a tag HTML IFram é usada para exibir uma página da Web adicional. |
| age\_of\_domain | o site vive por um curto período de tempo. |
| DNSRecord | existência de registro DNS para o domínio. |
| web\_traffic | popularidade do site. |
| Page\_Rank | o page rank. |
| Google\_Index | se o wesite é indexado pelo Google. |
| Links\_pointing\_to\_page | número de links que apontam para a página da Web. |
| Statistical\_report | registro do site em locais de rastreio de phishing como phishTank |
| Y = Result | resultado |

**2 Etapas do KDD – Scripts em Python**

**2.1 Seleção**

**Definição:** A etapa de seleção é a primeira etapa do KDD, é uma etapa muito importante, pois é nela que serão decididos quais os conjuntos de dados que serão relevantes para que sejam obtidos resultados com informações uteis.

possue valores 0 para seus atributos->

**URL\_Length, having\_Sub\_Domain, SSLfinal\_State, URL\_of\_Anchor, Links\_in\_tags, SFH, Redirect, web\_traffic, Links\_pointing\_to\_page**

Utilizamos todos os atributos listados na introdução

**2.2 Pré-processamento e Limpeza**

**Definição**: Acontece a limpeza dos dados e seleção de atributos. Nesta etapa informações ausentes, errôneas ou inconsistentes nas bases de dados devem ser corrigidas de forma a não comprometer a qualidade dos modelos de conhecimento a serem extraídos ao final do processo de KDD.

Convertemos os dados vindos no arquivo em inteiro, pois o mesmo está vindo como string.

**2.3 Transformação dos dados**

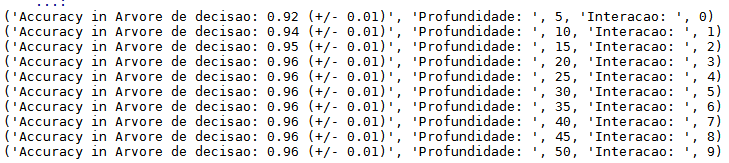
**Definição**: Analisa os dados obtidos da etapa anterior e os reorganiza de uma forma específica para que possam ser interpretados na etapa seguinte.

**2.4. Data Mining**

**Definição**: É onde tudo acontece, os dados depois de transformados serão lidos e interpretados. A mineração faz com que meros dados sejam transformados em informações, tais informações são indicadas através de força bruta, ou seja, lendo regra por regra e as interpretando.

**2.4.1 Arvore de Decisão Simples**

**Definição**: Árvores de decisão são um dos modelos mais usados em inferência indutiva. São métodos de aprendizado de máquinas supervisionado e não-paramétricos, muito utilizados em tarefas de classificação e regressão que representam funções como árvores de decisão. Uma árvore de decisão, em geral, em computação, são estruturas de dados que armazenam informações como um mapa dos possíveis resultados de uma série de escolhas relacionadas pelas regras em seus nós e a decisão tomada através do caminhamento a partir do nó raiz até o nó folha. Assim, permite que cenários sejam comparados com base em parâmetros como custos, probabilidades e benefícios. Podem pode ser usadas tanto para conduzir diálogos informais quanto para mapear um algoritmo que prevê a melhor escolha, matematicamente. Uma árvore de decisão por si só não é um aprendizado de máquina, mas o processo de construção automático de árvores de decisão a partir de um conjunto de dados o é. Para a construção destas árvores são usados algoritmos como o ID3, ASSISTANT e C4.5. Estas árvores são treinadas de acordo com um conjunto de treino (exemplos previamente classificados) e posteriormente, outros exemplos são classificados de acordo com essa mesma árvore.

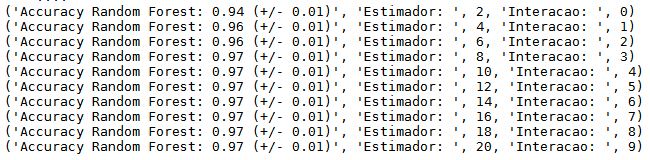


Para esse tipo de classificador, a melhor representação da imagem acima é:

|  |  |  |  |
| --- | --- | --- | --- |
| **Classificador Utilizado** | **Acurácia** | **Profundidade** | **Iteração** |
| tree.DecisionTreeClassifier | 0.96 | 20 | 3 |

**2.4.2 Random Forest**

**Definição**: O método Random Forest (Floresta Aleatória) é um algoritmo de aprendizagem de máquina supervisionada, flexível e é um dos algoritmos mais utilizados, devido à sua simplicidade e o fato de que pode ser utilizado para tarefas de classificação e também de regressão; que, na maioria das vezes, produz excelentes resultados, mesmo sem ajuste de hiperparâmetros. Funciona criando uma floresta de um modo aleatório, ou seja, uma combinação (ensemble) de várias árvores de decisão para obter uma predição com maior acurácia e mais estável. O algoritmo adiciona aleatoriedade extra ao modelo, quando está criando as árvores, pois ao invés de procurar pela melhor característica ao fazer a partição de nodos, ele busca a melhor característica em um subconjunto aleatório das características. Este processo cria uma grande diversidade, o que geralmente leva a geração de modelos melhores.



Para esse tipo de classificador, a melhor representação da imagem acima é:

|  |  |  |  |
| --- | --- | --- | --- |
| **Classificador Utilizado** | **Acurácia** | **Estimador** | **Iteração** |
| RandomForestClassifier | 0.967 | 8 | 3 |

**2.4.3 Rede Neural MLP**

**Definição**: A Perceptron Multicamadas (MLP) é um algoritmo padrão para qualquer aprendizado supervisionado no processo de reconhecimento de padrões e objeto de investigação em cursos de neurociência computacional e processamento distribuído em paralelo. São úteis na resolução de problemas estocásticos, e permite muitas vezes obter soluções aproximadas para problemas extremamente complexos. MLP é uma rede neural semelhante à Perceptron simples, mas com três ou mais camadas de neurônios em alimentação direta (uma camada de entrada e uma camada de saída com um ou mais camadas ocultas). O MLP é utilizado nos casos em que não há a possibilidade de uma única reta separar os elementos, pois assim, gera-se mais de uma reta classificadora. Tal tipo de rede é totalmente conectada composta por camadas de neurônios em que camada da rede tem uma função específica e elas são ligadas entre si, pelos seus nós, por sinapses com pesos, em um nó da camada seguinte. A camada de saída recebe os estímulos da camada intermediária e constrói a resposta. O aprendizado nesse tipo de rede é geralmente feito através do algoritmo de retro-propagação do erro, mas existem outros algoritmos para este fim, como a Rprop. A unidade fundamental de processamento de informação de uma rede neural artificial é um neurônio artificial com seus três elementos básicos (conexões de entrada, combinador linear e função de ativação).

Para esse tipo de classificador, a melhor representação da imagem acima é:

|  |  |  |  |
| --- | --- | --- | --- |
| **Classificador Utilizado** | **Acurácia** | **Profundidade** | **Iteração** |
|  |  |  |  |

**2.4.4 Comitê de Redes Neurais em Python**

**Definição**:

Para esse tipo de classificador, a melhor representação da imagem acima é:

|  |  |  |  |
| --- | --- | --- | --- |
| **Classificador Utilizado** | **Acurácia** | **Profundidade** | **Iteração** |
|  |  |  |  |

**2.5. Interpretação e Avaliação**

**Definição**: É onde as regras indicadas pelo processo anterior serão interpretadas e avaliadas. Após a interpretação poderão surgir padrões, relacionamentos e descoberta de novos fatos, que podem ser utilizados para pesquisas, otimização e outros.

**2.5.1 Gráficos de caixa (Boxplot)**

**2.5.2 Analise dos Resultados**

**3** **Aspectos Relevantes**

[**RRReferências**](#_56h4futan5g4)

* **xx**
* **xx**