Modelo de Validação de Dados Cadastrais para Utilização em Banco de Dados em Nuvem

Gabriel Bolson Dalla Favera, Henrique Gabriel Gularte Pereira

Centro Universitário Franciscano – Santa Maria – RS – Brasil

Curso de Sistemas de Informação – Trabalho Final de Graduação II

gabrielfavera@unifra.edu.br, henriquep@unifra.br

Abstract. *The need to validate untrusted data entry data has become a major challenge in the world of Internet programming. In this context, this article presents a computational model that seeks to validate a veracity of the information of users registered in a cloud database. To do this, we use the techniques of Automatic Learning, with the help of Weka software, providing answers according to the level of relevance of each set of input data. From the analysis of the results it is possible to observe that both the NaiveBayes algorithm and the J48 positive results are satisfactory in the classification and identification of false registrations.*

Resumo. *A necessidade da validação de dados de entrada advindos de fontes não confiáveis tem se tornado um grande desafio no mundo da programação para Internet. Nesse contexto, este artigo apresenta um modelo computacional que busca validar a veracidade das informações de usuários cadastrados em um banco de dados em nuvem. Para isso, é utilizada técnicas de Machine Learning, com a utilização conjunta do software Weka, fornecendo respostas de acordo com o nível de relevância de cada conjunto de dados de entrada. A partir da análise dos resultados é possível observar que tanto o algoritmo de NaiveBayes quanto o J48 apresentam resultados satisfatórios na classificação e identificação de cadastros falsos.*

1. Introdução

Em meio ao cenário atual do mundo globalizado mais de 2,4 bilhões de pessoas já usam algum serviço de computação em nuvem. Estima-se que até 2018, este número possa ser de aproximadamente 3,6 bilhões de pessoas [Sorrell 2014].

Sistemas de computação em nuvem podem fazer uso de várias formas de segurança. Dentre os mais utilizados destacam-se as técnicas de encriptação, forte controle de gestão e identificação de acessos, gestão de incidentes. Porém, de acordo com o portal de informações Statista (2016), cerca de 38% das empresas que usam computação em nuvem não adotam nenhuma medida especial de segurança.

Muitas dessas empresas armazenam dados de grande importância, porém não adotam nenhuma proteção adicional. Devido à grande quantidade de usuários, onde a chance de ataques é muito alta, e a tendência de ataques a servidores na nuvem vem

crescendo substancialmente [CERT.br, 2016], é essencial que estas empresas adotem algum tipo de segurança para garantir que seus dados não sejam violados.

Não somente empresas, mas usuários que fazem utilização de ambientes em nuvem correm riscos. Muitos utilizam estes sistemas para armazenar informações referentes a trabalhos de escola, trabalhos de faculdade, informações pessoais, além de documentos de trabalhos. A utilização destes sistemas permite uma melhor organização dos dados, sendo uma informação mais primitiva e fragmentada, anterior a interpretação, e informações, além da representação contida nos dados, facilitando acessos, provendo agilidade [Donela 2006].

Devido à popularização de sistemas baseados na Internet e consequentemente do aumento na utilização de bancos em nuvem para armazenamento de informações desses sistemas, é necessário a criação de técnicas e ferramentas capazes de permitir que estas aplicações validem os dados inseridos pelos usuários. É comum em aplicações para Internet que usuários utilizem dados cadastrais falsos ou tenham múltiplas contas. Grande parte dos bancos de dados em nuvem já fazem utilização de diferentes tipos de validadores de informações cadastrais, na maioria ineficiente ou com um baixo controle de informações. Dessa maneira, torna-se necessário autenticar dados cadastrais armazenados na nuvem com maior eficácia, possibilitando que medidas preventivas possam ser tomadas.

1.1. Objetivo

O presente trabalho tem como objetivo apresentar o desenvolvimento e análise de resultados de um modelo computacional capaz de identificar cadastros inconsistentes de usuários em sistemas de banco de dados em nuvem, utilizando *Machine Learning* e, classificando esses cadastros como verdadeiros ou falsos, utilizando a ferramenta Weka, que foi responsável pela mineração em todos os dados já cadastrados, treinando um modelo capaz de identificar registros com informações falsas, permitindo que medidas corretivas possam ser tomadas.

1.2. Estrutura do Trabalho

O trabalho está dividido em 7 seções. As Seções 2, 3 e 4 exibem as estruturas bibliográficas abordando respectivamente computação em nuvem, *Machine Learning* e o software Weka. A Seção 5 expõe os trabalhos relacionados. A Seção 6 aborda a metodologia utilizada assim como seus métodos de validação das informações. A Seção 7 os resultados encontrados e a Seção 8 mostra a conclusão.

2. Computação em Nuvem

Mell e Grance (2011) definem computação em nuvem como um modelo que permite múltiplos acessos para compartilhamento de diferentes recursos computacionais. Sendo o funcionamento de banco de dados como servidores, diferentes tipos de armazenamentos, aplicações, serviços, entre outros. Permite ser acessado de maneira ágil e de diferentes localizações.

Segundo Velte et al. (2011) a computação em nuvem é uma maneira de cortar custos operacionais ao invés de manter *datacenters* funcionando, permitindo que profissionais de Tecnologia da Informação (TI) foquem em projetos estratégicos. Para Sosinsky (2011) existem duas diferentes classes: os com bases para modelos de implementação e os com base para modelo de serviço.

Aplicações em nuvem são inerentemente distribuídos e, portanto, eles são necessariamente entregues através de uma rede. As maiores aplicações podem envolver milhões de usuários, e o método de transporte é geralmente a Internet [Yelury e Castro-Leon 2014]. A ideia de disponibilizar aplicações de software sobre uma rede surgiu com o conceito de compartilhamento de tempo (*time sharing*) durante os anos de 1960 e 1970. A ideia evoluiu para aplicações hospedadas entre os anos de 1980 e 1990 [Cusumano 2010].

Para Mell e Grance (2011), infraestruturas computacionais classificadas como computação em nuvem devem apresentar algumas características como: serviço sob demanda em que o usuário possa obter recursos computacionais, como tempo de processamento ou armazenamento; amplo acesso à rede, facilidade de acesso por diferentes recursos computacionais; *pooling* de recursos, prover recursos computacionais a atender vários consumidores através do modelo multiusuário (consiste em atender várias requisições de diferentes usuários), com uma ampla variedade de recursos físicos e visuais de maneira dinâmica conforme a demanda de consumidor; rápida elasticidade, em que recursos podem ser adquiridos de maneira rápida e elástica, em algumas situações automaticamente; e serviço medido, onde sistemas em nuvem controlam e otimizam recursos automaticamente, alavancando a capacidade de medição adequado ao tipo de serviço.

3. *Machine Learning*

Um sistema que trabalha em *Machine Learning* (Aprendizagem de Máquina) tem como função estudar e compreender diferentes processos de entrada e saída, sem a necessidade de uma programação especifica [Witten et al. 2005].

Originalmente *Machine Learning* surgiu juntamente com a Inteligência Artificial (IA) nos anos de 1940, no início dos primeiros computadores. Dessa maneira, teve a necessidade de um sistema que tomasse suas próprias decisões de acordo com o que já foi aprendido pelo sistema e pela melhor alternativa [Russell e Norvig 2003]. O *Machine Learning* teve suas primeiras versões em 1986 [Langley 2011].

No *Machine Learning* existem três tipos de controles de tarefas, sendo elas: não supervisionado, supervisionado e semissupervisionado. Algoritmos de *Machine Learning* não supervisionados ou não rotulados são modelos de aprendizado de máquina capazes de serem utilizados em situações onde não possuem padrões, relações, regularidades ou categorias pré-definidas, sendo utilizados para reconhecer padrões de grupos com prioridades similares [Nogueira 2009].

Algoritmo supervisionado é aquele onde os dados são previamente classificados, sendo que em seu conjunto de treinamento, a cada classe é atribuída um rótulo. O sucesso da aprendizagem de classificação pode ser julgado ao se comprar os resultados aprendidos pelo modelo contra um conjunto independente de dados de teste para os quais são conhecidas as verdadeiras classificações, mas não colocados à disposição da máquina. A

taxa de sucesso em dados de teste dá uma medida objetiva de quão bem o conceito que foi estudado [Witten et. al. 2011].

Os algoritmos de aprendizado considerados semissupervisionado são divididos em: visão única e visão múltipla. Os algoritmos de visão única normalmente utilizam apenas um classificador, podem ser subdivididos em: transdutivos no qual são variações de Expectation Maximization (EM), algoritmos com aprendizagem e algoritmos de *clustering*. Já os algoritmos de visão múltipla utilizam dois ou mais classificadores. Um dos modelos que podem ser utilizados é o de Co-Training, que utiliza pelo menos dois classificadores independentes, geralmente um rotulado (previamente classificado) e melhora o desempenho do algoritmo a medida que novas descobertas vão sendo efetuadas pelo classificador não rotulado [Witten et al. 2011].

Uma ampla variedade de algoritmos de Machine Learning é disposta para autenticar informações. Algoritmos Random tem como base a árvore de decisões em que as melhores escolhas são realizadas de acordo com aqueles que apresentam as melhores respostas, além de que todas as alternativas são testadas. Já algoritmos Bayesianos tem como objetivo calcular a probabilidade e predizer a classe mais provável, também conhecido como classificação estática. Algoritmos Functions utilizam-se de *clustering* para fornecer funções de base e aprender tanto com regressão lógica quanto regressão linear [Castro e Ferrari 2016].

Algumas aplicações para *Machine Learning* incluem: adaptação para websites, anatomia computacional, visão computacional para o reconhecimento de objetos, detecção de fraude em cartão de crédito, recuperação de informação, detecção de fraude na Internet, percepção de máquina, motores de buscas, reconhecimento de voz e escrita manual, entro outras aplicações [Smola e Vishwanathan 2008].

Diferentes softwares de *Machine Learning* já são utilizados, alguns *Open-Source* e outros com licença comercial. Os softwares *Open-Source* mais comuns são, Weka, OpenCV, OpenNN, TensorFlow, além de alguns outros modelos. Para softwares comercias, Google Cloud Machine Learning Platform, Microsoft Azure Machine Learning, Oracle Data Mining, MATLAB, Wolfram Alpha, entre outros.

4. Weka

Weka é uma ferramenta de *Knowledge Discovery in Databases* ou descoberta de conhecimento em base de dados - *KDD* que contém uma série de diferentes algoritmos para preparação de dados, aprendizagem de máquina (*Machine Learning*) e validação de resultados [Silva 2004].

A Weka é um ambiente que oferece uma ampla variedade de algoritmos para diferentes tipos de análises de dados. Tendo como base atual de linguagem de programação o Java, novos algoritmos podem ser utilizados visando o tipo de análise a ser feita. Dessa forma, novos dados para análise podem ser inseridos de maneira simples com instruções aceitas pelo Weka, sendo analisados e apresentadas suas respostas de maneira intuitiva, alguns algoritmos disponibilizam os resultados por meio de porcentagens e tabelas [Witten et. al. 2011].

A Weka é capaz de servir como base para diferentes tipos de utilização. É possível utilizar algoritmos não supervisionados disponibilizados para diferenciar padrões

textuais, e padrões de palavras, bem como algoritmos supervisionados para realizar classificações [Witten et. al. 2011].

5. Trabalhos Relacionados

Para o desenvolvimento da proposta e da metodologia, foi realizada uma pesquisa bibliográfica acerca do tema da identificação de informações inválidas ou incorretas em sistemas computacionais utilizando técnicas de inteligência artificial. A seguir serão apresentados os trabalhos mais relevantes.

Kater e Jäschke (2016) apresentam uma técnica para detecção de cadastros considerados como *spam*. Para isso eles utilizam *Machine Learning*, e um conjunto de 177 características obtidas durante a etapa de cadastro do usuário na aplicação. Dentre as características analisadas pelo modelo proposto é possível encontrar informações básicas como nome e sobrenome, e também cálculos mais complexos como a quantidade de letras da primeira fila do teclado digitadas em sequência em um campo do formulário de cadastro. O trabalho se utiliza da ferramenta Weka para realizar a análise dos dados e de uma técnica de validação cruzada para apurar os resultados. Os resultados mostram um baixo número de falsos negativos (identificação incorreta de resultado correto) e falsos positivos (identificação incorreta de resultado falso), menos de 1% pelo teste de mais de 100000 validações, apresentando bons resultados e permitindo a detecção de *spam*. Dessa maneira tornou-se possível bloquear os falsos cadastros de usuários em tempo hábil.

Xiao et al. (2015) apresentam uma técnica de identificação de usuários com cadastros múltiplos. A metodologia fez uso de algoritmos supervisionados de *Machine Learning*. Dentre as características analisadas é possível destacar informações provenientes do usuário, como nome e sobrenome, e-mail, nível de ensino ou empresa. A metodologia apresentada tem sido utilizada em produção na empresa LinkedIn e, até a data da publicação do artigo, cerca de 250.000 cadastros falsos haviam sido detectados utilizando essa técnica.

Geng et al. (2009) busca detectar *spam* em um ambiente *web*. Para isso utiliza algoritmos semissupervisionados buscando melhorar o desempenho dos classificadores, com integração de auto treinamento. Os algoritmos utilizam *Machine Learning* para treinamento dos dados a partir de um conjunto reduzido de dados rotulados. Para avaliação e aperfeiçoamento do algoritmo proposto, foram utilizados mais de 77.9 milhões de páginas disponibilizadas. Os resultados apresentados mostram que é possível utilizar métodos de aprendizagem semissupervisionados para melhorar a taxa de acerto dos classificadores.

Dentre os trabalhos relacionados, o presente trabalho apresentou respostas com um nível de satisfatório ao que foi proposto, muito similar ao Kater e Jäschke, com baixa quantidade de falsos negativos e falsos positivos. Referente aos demais trabalhos abordados foram utilizados algumas similaridades de avaliações, além de utilizar o algoritmo de validação, NaiveBayes, a fim de buscar resultados mais próximos de 100% (cem por cento) de precisão.

6. Metodologia

Para o desenvolvimento do modelo, foi realizado um estudo sobre *Machine Learning*, juntamente com a geração do modelo de aprendizagem supervisionado. Com isso, foi realizada a coleta e triagem dos cadastros do banco de dados em nuvem, tendo todos os cadastros sido previamente classificados em cadastros verdadeiros e cadastros falsos. Após essa etapa, foram realizados testes cruzados para a avaliação do modelo, obtendo assim resultados para validações. Nessa seção serão descritos os passos que foram realizados durante a execução do trabalho.

6.1. Obtenção dos dados

Os dados utilizados na execução do trabalho e no treinamento do modelo são oriundos de uma base de dados com cadastros reais provindos de uma instituição de ensino superior. Os dados foram exportados de um banco de dados DB2 e convertidos para o formato de Comma-separeted values (CSV). Um exemplo de registro originalmente obtido nesse formato pode ser visto no Quadro 1.

Quadro 1. Exemplo de Registro no Formato CSV oriundo do DB2.

```
XXX XXX; 0309522XXXX; XXXrolim7@gmail.com; 201211710; 07.05.1993;
Divisa Alegre; Minas Gerais
```

Os dados presentes no arquivo, em ordem, são: nome, CPF, e-mail, matricula, data de nascimento, cidade e Estado. Além desses dados, também foram importados a data de último acesso ao sistema, data de cadastramento ao sistema e telefone.

Esses dados foram obtidos no formato utilizado pelo banco e para serem utilizados pelo Weka foi necessário implementar um conjunto de ferramentas capaz de converter e normalizar esses dados. Para isso, optou-se pela utilização de *scripts* em Python. O Quadro 2 apresenta as linhas relativas a importação do arquivo CSV em uma das ferramentas desenvolvidas.

Quadro 2. Exemplo de aplicação Python que carrega o arquivo CSV e gera uma lista ordenada a partir do primeiro campo convertido para maiúsculo.

```
import csv
arquivo = open("ConjuntoDeTreinamento.txt", "r", newline='')
leitor = csv.reader(arquivo, delimiter = ",", quotechar = '"')
lista = []
for linha in leitor:
   uplinha = linha
   uplinha[0] = linha[0].upper()
   lista.append(uplinha)
lista.sort()
```

Após a normalização desses dados, eles foram carregados em uma base de dados MySQL, que foi utilizada para permitir que o Weka tive acesso a essas informações diretamente com o objetivo de treinar o modelo e realizar as consultas. A partir da

obtenção dos dados pelo Weka, foi possível realizar o treinamento de cada um dos conjuntos de dados, utilizando os diferentes algoritmos disponíveis na ferramenta. A Figura 1 apresenta o treinamento do conjunto de dados nome utilizando o algoritmo NaiveBayes. Esse processo foi repetido para cada um dos conjunto de dados do modelo e para cada um dos algoritmos disponíveis na Weka com o objetivo de identificar os que apresentavam os melhores resultados.

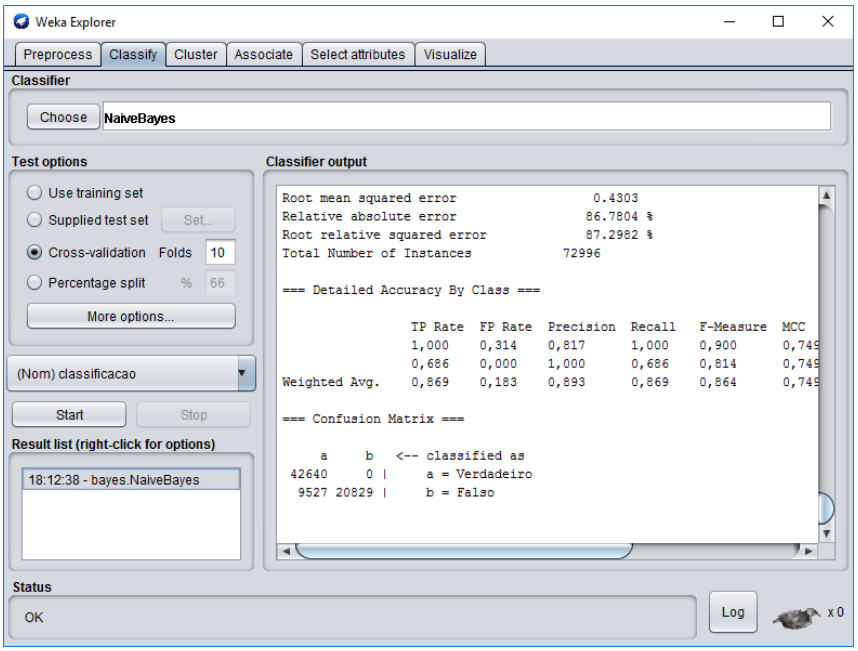


Figura 1. Tela de treinamento do modelo na Weka.

6.2. Tipo de dados

Os principais tipos de entradas de dados para a análise são nomes, datas de nascimento, datas de cadastramento, datas de último acesso, CPFs, cidades, Estados, e-mails e telefones. Dessa forma, os dados que foram analisados forneceram uma resposta em porcentagem, apresentando a resposta mais precisa.

6.3. Conjunto de Treinamento

Para quantificar a chance de os dados serem verdadeiros foram implementadas instruções para verificação. Os conjuntos foram treinados de diferentes maneiras, individualmente, agrupados por afinidade conforme descrito no modelo proposto, e todos os dados em conjunto, buscando com maior exatidão melhores resultados. Foram utilizados os algoritmos disponíveis na ferramenta Weka 3.8 para realizar o treinamento do modelo.

O conjunto do nome verifica através de uma base de dados, contendo variadas nomenclaturas, igualdades ou similaridades de palavras. A utilização de algoritmos supervisionados permitirá uma análise mais precisa, não descartando a ocorrência de nomes mais diferenciados.

O conjunto do CPF é treinado utilizando algoritmos supervisionados, através do cálculo por padrão e comparando a uma base de dados contendo padrões referentes a cada Estado. Como análise inicial, é feito por um cálculo que identifique o padrão de CPF. O

Estado em que foi informado é analisado junto ao CPF, determinado o padrão a ser avaliado.

No conjunto cidade e Estado serão comparados em conjunto, conforme a cidade abordada deve pertencer a determinado Estado. Algoritmos de comparações simples podem validar as informações. As informações serão pertencentes ao país analisado.

O conjunto e-mail fornece uma variedade de características, sendo a presença de ao menos um caractere antes e após a **@** (arroba), necessita da existência no mínimo de um ponto com dois caracteres após.

No conjunto de data de nascimento é verificado a consistência referente a questões de tempo. As datas serão divididas em 3 conjunto (dia, mês e ano), sendo analisados possíveis discrepâncias em relação aos dados de treinamento.

Para validação de número de telefone, é utilizada uma base de dados que pode ser analisada de acordo com cidade e Estado em que está vinculado. Questões de números de celular e telefones fixos podem ser diferenciados, definindo padrão de números para diferentes cidades e estados.

As datas de cadastramento, último acesso e atual permitem um controle de uso por parte de cada usuário no sistema nuvem. As datas de último acesso podem ser comparadas com a atual, apresentando tempo de uso médio, caso tenha uma grande variância de tempo desde o último acesso podemos determinar como falso usuário ou inativo.

6.4. Estratégia de Avaliação

Para conduzir a avaliação de falsidade de dados cada conjunto de treinamento foi avaliado separadamente, atribuindo uma porcentagem de falsidade perante as informações. Cada conjunto treinado possui uma determinada relevância, variando de níveis de 1 (um) à 5 (cinco), sendo 1 de pouca importância e 5 de grande importancia.

O conjunto nome apresenta nível 2, devido à dificuldade de avaliar uma grande quantidade de nomes estrangeiros ou exclusivamente diferentes. A base de dados é treinada a reconhecer fonemas e uma grande variedade de nomes brasileiros. Pode apresentar situações de o nome cadastrado ser estrangeiro.

O conjunto CPF apresenta nível 4, devido a padronizações matemáticas e por estado, apresentado outras características de CPFs válidos. A validação pode ser eficiente, podem apresentar CPFs validos que foram regidos pelos padrões de avaliação.

O conjunto de cidade e Estado é o de maior importância tendo nível 5. Esse conjunto pode ser facilmente validado verificando na base de dados e confirmando se a cidade pertence ao Estado, só serão permitidos campos verdadeiros e existentes. Esse conjunto quando validado verdadeiro fornece informações para CPF e telefone.

O conjunto telefone é de nível de relevância 3. Telefones fixos apresentarão resultados mais exatos com relação a região informada, sendo que cada cidade e Estado apresentam padrões. Telefones celulares podem variar, como o caso de pessoas que viajam entre diferentes cidades e Estados.

O conjunto e-mail é de relevância 2. E-mail pode ser facilmente validado desde que contenham uma @ com pelo menos um caractere antes e depois, tendo inclusive um

ponto com ao mínimo um caractere após a @. Apesar da facilidade e-mails falsos podem ser facilmente criados, não apresentando assim uma relevância tão grande.

O conjunto de data de nascimento tem nível 2. As datas são analisadas considerando dia, mês e ano. Isso possibilita comparar dia e mês, impossibilitando datas inválidas. O modelo utilizará as datas de nascimento já cadastradas para identificar possíveis desconexões em relação aos dados que estão sendo validados.

O conjunto de data de inscrição e último acesso é de nível 1. A data de último acesso pode ser comparada com a atual, que conforme o tempo em uso pode determinar usuário falso ao até mesmo inativo, ou seja, que usou para teste e não usou mais por ineficiência ou desinteresse. As datas de cadastramento e último acesso podem oferecer o tempo em que foi devidamente usado.

A avaliação dos conjuntos de análise é feita por média ponderada, em que é atribuído uma porcentagem para cada nível apresentado nos conjuntos. Dessa maneira, é possível quantificar a importância referente a cada conjunto. Os conjuntos nome, CPF, cidade e Estado, telefone, e-mail, data de nascimento e data de inscrição e último acesso obterão a porcentagem referente ao nível estipulado, respectivamente, 11%, 21%, 26%, 16%, 11%, 10% e 5%, formando assim uma avaliação de 100% referente a todos os conjuntos.

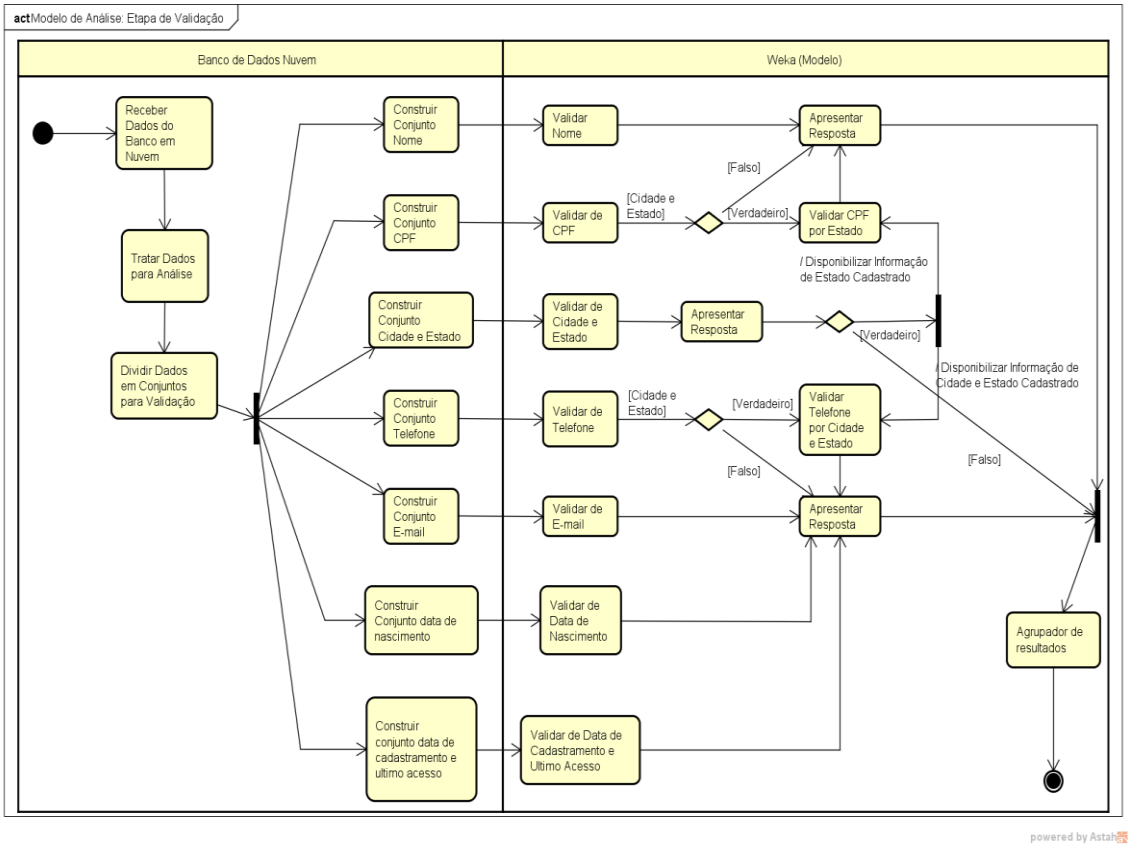


Figura 2. Representação do Modelo de Validação

A Figura 2 representa o diagrama de atividades do modelo de validação, que consiste em autenticar as informações que são obtidas em bases de dados da nuvem, sendo predominantemente informações de cadastros de usuários, ainda não classificados. Dos dados que são obtidos, são feitos tratamentos para análise e divididos de acordo com o

modelo proposto e implementado na Weka. As informações são divididas em conjuntos, sendo validadas de acordo com cada conjunto de treinamento, em alguns casos as informações de cidade e estado são disponibilizadas para complementar outras análises.

A porcentagem obtida nos conjuntos avaliados é utilizada para estimar a chance dos dados cadastrados serem falsos. Dos valores obtidos abaixo de 50% o usuário cadastrado constituirá definitivamente falso, valores de 50% até 75% apresentará como usuário possivelmente falso, de 75% a 85% constituirá possivelmente verdadeiro e acima de 85% definitivamente verdadeiro. Com os resultados já adquiridos, medidas preventivas poderão ser tomadas por parte do administrador do banco de dado na nuvem.

A validação cruzada é uma técnica utilizada para avaliar a capacidade de generalização de um modelo, permitindo comparar informações de diferentes situações buscando um certo nível de similaridade. Essa técnica é vastamente usada em aplicações que buscam realizar a predição de dados.

A técnica de validação cruzada consiste em separar um conjunto finito e classificado da base de treinamento e utilizar esse conjunto de dados para calcular a taxa de acerto do modelo. É possível calcular essa taxa de acerto ao se comparar as respostas dadas pelo modelo e os rótulos de classificação já conhecidos, sendo assim possível calcular a taxa de acerto, o número de falsos positivos e o número de falsos negativos. Buscou-se que o número de falsos positivos e falsos negativos tendesse a zero, para que seja possível obter uma taxa de acerto de 100%.

A base de treinamento consistiu de 72640 (setenta e dois mil seiscentos e quarenta) registros, sendo 42640 (quarenta e dois mil seiscentos e quarenta) registros conhecidamente verdadeiros e 30000 (trinta mil) registros falsos.

7. Resultados

Para a criação do modelo de treinamento foi utilizada uma base de dados com cadastros de um sistema web da Universidade Federal de Santa Maria, que continha 42640 dados cadastrados verdadeiros. Os dados foram então separados e tratados com o objetivo de alimentar o modelo e permitir a classificação.

Os registros falsos foram gerados a partir de uma aplicação em Python, em conjunto com a biblioteca Faker, a fim de objetivar a criação de entradas artificiais para simular os registros possivelmente falsos em um banco de dados real. No total, 30000 dados cadastrais falsos foram gerados para avaliação, sendo que cada um desses registros possuía o mesmo número de informações que os dados verdadeiros, e na maioria das situações sendo muito similares aos verdadeiros, fazendo com que esses registros fossem praticamente indistinguíveis dos dados verdadeiros.

Os registros foram armazenados em um banco de dados MySQL com conexão direta ao Weka, permitindo uma busca direta de cadastros, tendo assim maior rigorosidade nas avaliações.

Dentre os vários algoritmos disponíveis na Weka, o que apresentou os melhores resultados foi o NaiveBayes, do grupo dos algoritmos Bayesianos, e o algoritmo J48, do grupo dos algoritmos Random. Outros algoritmos testados do tipo Random, foram o RandomForest e o RandomTree, já do tipo Functions, foi testado o algoritmo Logistic.

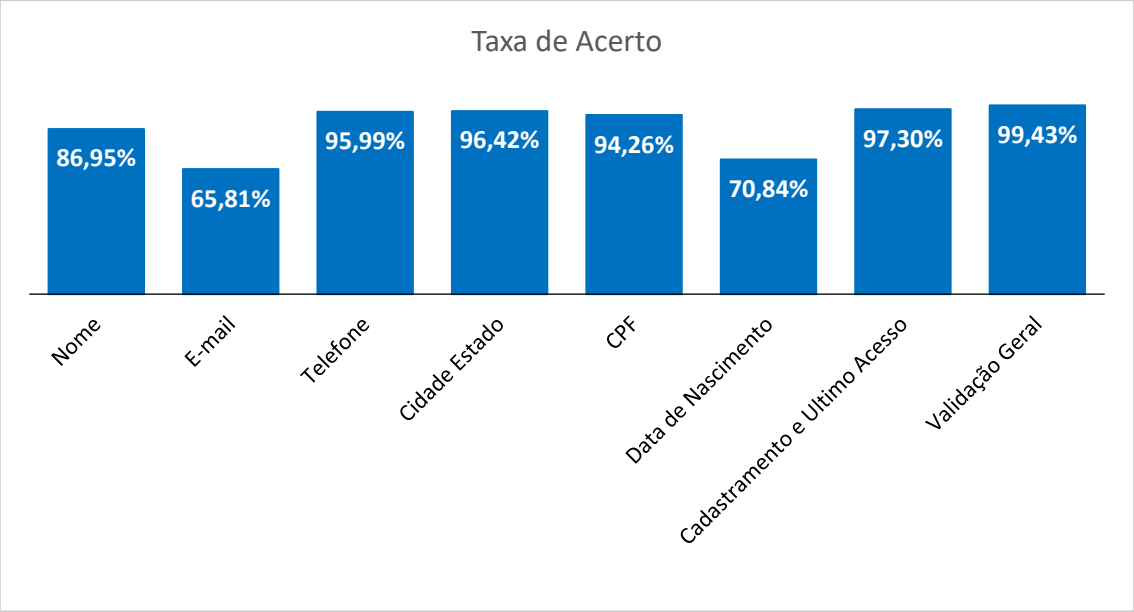
Com exceção do NaiveBayes e do J48, todos os demais algoritmos testados tiveram grandes dificuldades em classificar os dados.

Na maioria dos casos, o problema enfrentado foi a grande quantidade de dados a serem validados, ocorrendo estouro de memória ou ineficiência de resultados. Para que fosse possível validar com algoritmos do tipo Random foi necessário reduzir a base para menos de 10000 (dez mil) registros, o que ocasionou uma taxa de apenas 65% (sessenta e cinco) de acertos. Já para a execução dos algoritmos Functions foram necessárias uma redução para 5000 (cinco mil) dados, tendo como resultado uma taxa de 55% (cinquenta e cinco) de acertos. Para a reduzir a quantidade de cadastros, foi feito uma escolha aleatória utilizando a própria consulta SQL. Os resultados desses algoritmos tornaram ineficiente suas utilizações.

7.1. NaiveBayes

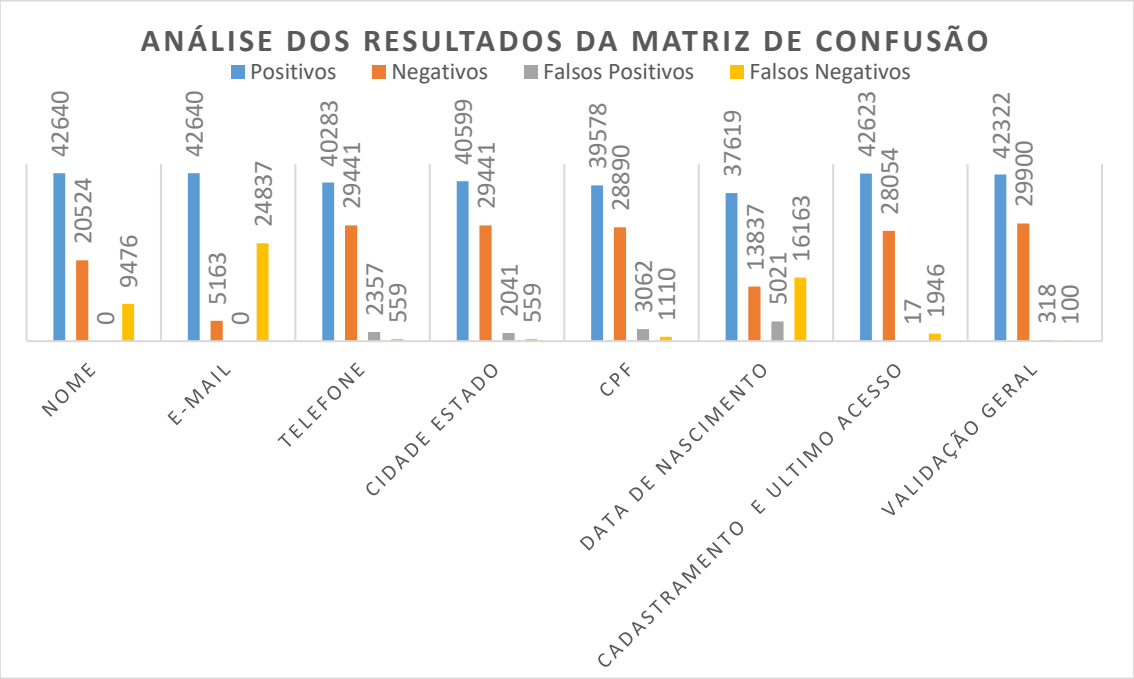
Durante as validações o algoritmo NaiveBayes apresentou bons resultados na maior parte dos testes de conjuntos e na validação geral. No caso de e-mail e data de nascimento os resultados foram abaixo do esperado. Nos conjuntos de telefone e cidade estado, os resultados apresentados foram inferiores aos resultados encontrados com o algoritmo de validação J48.

Gráfico 1. Resultados encontrados com o algoritmo NaiveBayes.



No Gráfico 1, são apresentados os resultados de treinamento encontrados de acordo com cada conjunto e juntamente com a avaliação geral em que contém todos os dados avaliados anteriormente em um único treinamento. O resultado informa a porcentagem de acerto do algoritmo em cada conjunto de dado. No conjunto de e-mail apresentou baixos resultados devido à grande opção variedade e a não padronização, sendo que para que um e-mail seja verdadeiro ele deve ter um @ com ao menos alguma palavra antes e depois com um ponto e uma palavra depois. Isso permite que uma ampla variedade de opções possa ser criada.

Gráfico 2. Matriz de confusão do NaiveBayes.

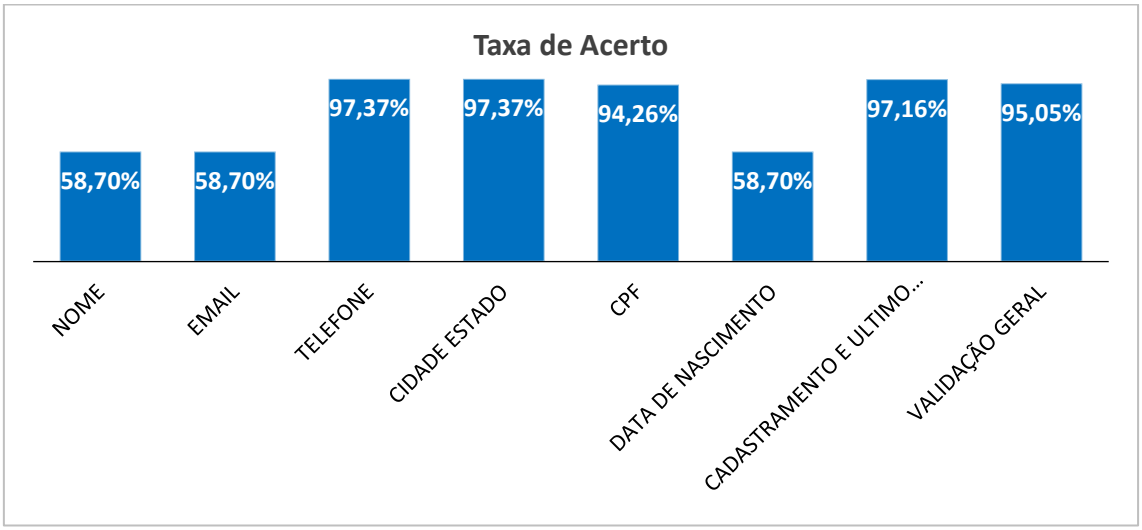


No Gráfico 2 é apresentado a matriz de confusão de cada conjunto e da validação geral. Nela são apresentados a quantidade de positivos, negativos, falsos positivos e falsos negativos. Nos conjuntos de nome e e-mail não apresentaram falsos positivos, tendo todos os positivos validados corretamente. As validações que utilizaram o algoritmo NaiveBayes apresentaram resultados superiores na maioria dos conjuntos em relação ao algoritmo J48.

7.2. J48

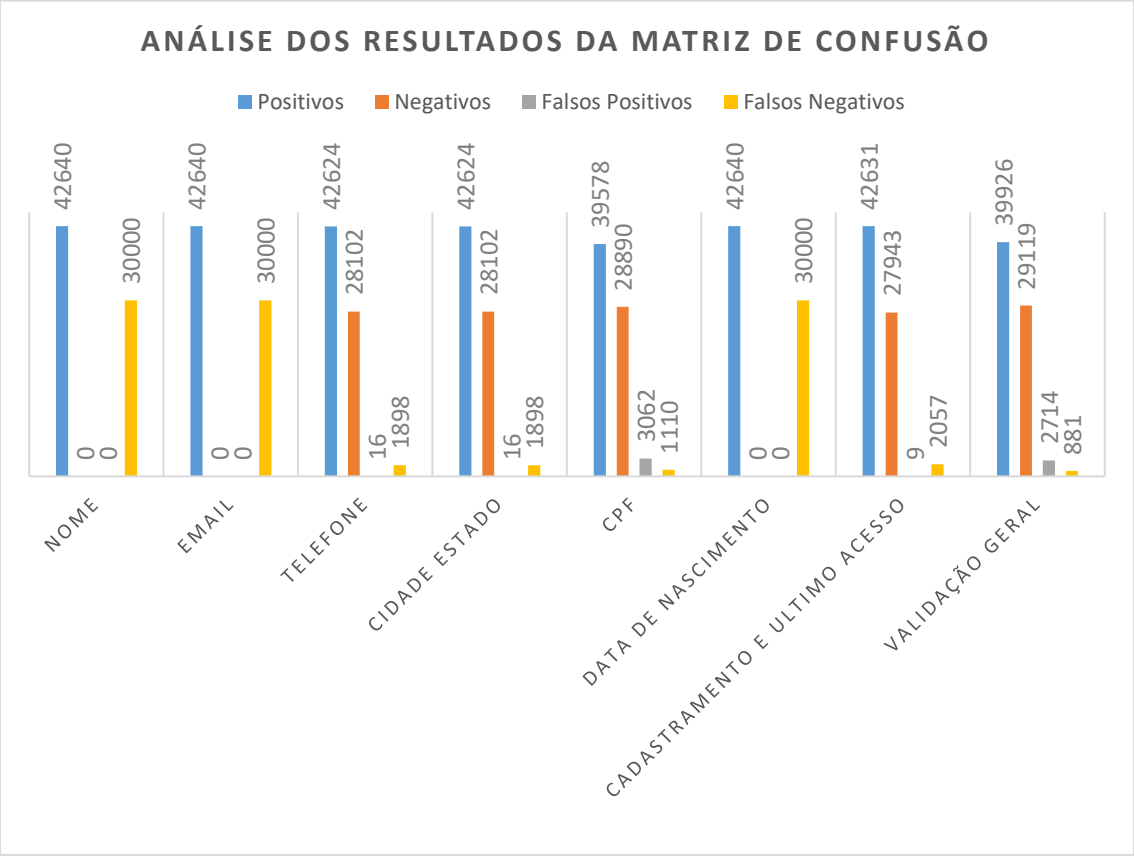
O algoritmo de validação J48, apresentou bons resultados em quatro conjuntos, sendo eles: cidade e Estado; CPF; data de cadastramento e último acesso; e telefone. Ele também apresentou bom resultado na validação geral.

Gráfico 3. Resultados encontrados com o J48.



No Gráfico 3, é apresentado os resultados encontrados em cada conjunto validado e a validação geral. Nos conjuntos de telefone, e cidade e Estado, os resultados apresentados foram melhores que na validação com o algoritmo NaiveBayes, e no caso de CPF o resultado foi o mesmo. Na data de cadastramento e ultimo o resultado com o algoritmo J48 ficou com apenas 0,14% menor que com o algoritmo Bayesiano.

Gráfico 4. Matriz de confusão do J48.



No Gráfico 4, é apresentado a matriz de confusão dos conjuntos e da validação geral. Consta a quantidade de positivos, negativos, falsos positivos e falsos negativos. Nos algoritmos que tiveram bons resultados o número de falsos positivos foi muito inferior a mesma quantidade quando validado no algoritmo Bayesiano, mas a quantidade de falsos negativos foi um pouco maior. Nos conjuntos que não apresentaram um bom resultado o maior problema foi a quantidade de falsos negativos, em que apresentava o número máximo de erros ou em comparação a validação com o NaiveBayes apresentou piores resultados.

O algoritmo permitiu o teste dos outros conjuntos, mas em três casos os resultados apresentaram que todos os dados analisados eram verdadeiros, obtendo uma média de menos de 60% de acertos devido a quantidade de dados verdadeiros e falsos no banco. O algoritmo J48 apresentou bons resultados sempre que tinha mais de três informações durante as análises.

8. Conclusão

O modelo proposto buscou identificar cadastros inconsistentes em sistemas de banco de dados em nuvem, utilizando *Machine Learning* e, classificando esses cadastros como verdadeiros ou falsos. As autenticações dos dados cadastrados foram situadas na utilização da Weka, tendo empregado alguns métodos do *Machine Learning* que buscou prever as possíveis falsas nomenclaturas. Outra característica foi a utilização de algoritmos supervisionados permitindo melhor controle de informações mais diferenciadas, não descartando e reavaliando.

Por meio dos resultados obtidos, foi possível determinar que os melhores algoritmos para validação foram o NaiveBayes e J48, em que tiveram bons resultados e que em alguns conjuntos validados um superava o outro, atingindo melhores resultados. Os resultados finais obtidos através da validação em conjunto, apresentaram resultados inferiores a validação geral, em que valida todas as informações, apresentaram respectivamente os resultados de 84,38%, variando os algoritmos a fim da melhor resposta, e 99,43%, utilizando o algoritmo NaiveBayes. Dentre os resultados alcançados, foi possível apresentar que validar todos os dados de maneira conjunta, sem separações, apresenta melhores respostas, além de apresentar uma maior quantidade de informações a fim de encontrar um padrão.

Para trabalhos futuros o desenvolvimento de uma biblioteca para a utilização em diferentes linguagens web ou de desenvolvimento de software pode ser aplicado. Isso permitirá que as validações sejam efetuadas de maneira prévia a confirmação do cadastro, dessa maneira a biblioteca disponibilizará uma porcentagem de autenticidade das informações inseridas pelo cliente para que o administrador do software ou página web possa determinar como um cadastro falso ou verdadeiro.

Referências

Castro, L. N.; Ferrari, D. G. (2016) Introdução à Mineração de Dados: Conceitos Básicos, Algoritmos e Aplicações. 1ª ed., Saraiva.

CERT.br. (2016) CERT.br aponta aumento de notificações de ataques a servidores Web. http://www.nic.br/noticia/releases/cert-br-aponta-aumento-de-notificacoes-de-ataques-a-servidores-web/, acessado em 18/03/2016.

Cusumano, M. (2010) Cloud Computing and SaaS as new computing platforms. Commun. ACM, New York, NY, USA.

Donela, D. (2006) Da privacidade à proteção de dados pessoais. Renovar, 1ª ed., Rio de Janeiro, Brasil.

Geng, G.; Li, Q.; Zhang, X. (2009) Link based small sample learning for web spam detection. In Proceedings of the 18th international conference on World wide web (WWW '09). ACM, New York, NY, USA, 1185-1186.

Kater, C.; Jäschke, R. (2016) You shall not pass: detecting malicious users at registration time. In Proceedings of the 1st International Workshop on Online Safety, Trust and Fraud Prevention (OnSt '16). ACM, New York, NY, USA, Article 2, 6 pages.

Langley, P. (2011) The Changing Science of Machine Learning. Springer. Springer, 1ª ed., Computer Science and Engineering, Arizona State University, P.O. Box 87-8809, Tempe, AZ 85287, USA.

Mell, P.; Grance, T. (2011) The NIST Definition of Cloud Computing. National Institute of Standards and Technology, [S.I.], p.7.

Nogueira, B. M. (2009) Avaliação de métodos não-supervisionados de seleção de atributos para Mineração de Textos. USP, São Carlos.

Pereira, H. G. G. (2012) Uma Arquitetura Para a Utilização de Computação nas Nuvens nos Ambientes de Computação Pervasiva. UFSM, Santa Maria, RS, Brasil.

Russell, S. J.; Norvig, P. (2003) Artificial Intelligence: a Modern Approach. Prentice Hall, 2ª ed., Upper Saddle River, New Jersey, USA.

Silva, M. (2004). Mineração de dados-conceitos, aplicações e experimentos com weka. Livro da Escola Regional de Informática Rio de Janeiro-Espírito Santo. Rio Janeiro: SBC.

Smola, A.; Vishwanathan, S. V. N. (2008) Introduction to Machine Learning. Cambridge University Press, 1ª ed., Cambridge, United Kingdom.

Sorrell, S. (2014) Consumer Cloud, There's No Limit. Juniper Research. Basingstoke, Hampshire

Sosinsky, B. (2011) Cloud Computing Bible. In: Defining Cloud Computing. Wiley Publishing, 1ª ed., Indianapolis, India.

Statista. (2016) Which of the following controls have you implemented to mitigate the new or increased risks related to the use of cloud computing? http://www.statista.com/statistics/258793/controls-taken-against-cloud-computing-security-risks/, acessado em 18/04/2016.

Velte, A. T.; Velte, T. J.; Elsenpeter, R. (2011) Computação em Nuvem: Uma Abordagem Prática. Alta Books, 1ª ed., Rio de Janeiro, Brasil.

Witten, I. H.; Frank, E.; Hall, M. A. (2011) Data Mining: Pratical Machine Learning Tools and Techniques. Elsevier, 3ª ed., Burlington, MA, USA.

Xiao, C.; Freeman, D. M.; Hwa, T. (2015) Detecting Clusters of Fake Accounts in Online Social Networks. In Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security (AISec '15). ACM, New York, NY, USA, 91-101.

Yelury, R.; Castro-Leon, E. (2014) Building The Infrastructure For Cloud Security: A Solution View. Apress Open, 1ª ed., Oregon, USA.

Anexo I – Estruturas de Tabelas

```sql
--
-- Estrutura da tabela cidade
--

CREATE TABLE cidade (
  `cidade` varchar(255) NOT NULL,
  `estado` varchar(255) NOT NULL,
  `ddd` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estrutura da tabela modelo
--

CREATE TABLE modelo (
  `idModelo` int(11) NOT NULL,
  `nome` varchar(100) DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  `telefone` varchar(20) DEFAULT NULL,
  `cpf` varchar(15) DEFAULT NULL,
  `cidade` varchar(50) DEFAULT NULL,
  `estado` varchar(50) DEFAULT NULL,
  `dt_nasc` date DEFAULT NULL,
  `dt_cadastramento` date DEFAULT NULL,
  `dt_ultimoacesso` date DEFAULT NULL,
  `classificacao` varchar(50) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Anexo II – Relação das consultas utilizadas na Weka para obtenção de cada conjunto de dados

Conjunto nome:

```
SELECT nome, classificacao FROM modelo
```

Conjunto e-mail:

```
SELECT email, classificacao FROM modelo
```

Conjunto telefone:

```
SELECT telefone, cidade, estado, classificacao FROM modelo
```

Conjunto cidade Estado:

```
SELECT cidade, estado, classificacao FROM modelo
```

Conjunto CPF:

```
SELECT cpf, estado, classificacao FROM modelo
```

Conjunto data de nascimento:

```
SELECT cast(dt_nasc as char), classificacao FROM modelo
```

Conjunto data de cadastramento e ultimo acesso:

```
SELECT cast(dt_cadastramento as char), cast(dt_ultimoacesso as char), classificacao FROM modelo
```