UNIVERSIDADE FEDERAL DO ACRE - UFAC
Programa de Pós-Graduação em Ciência da Computação - PPgCC

EDKALLENN SILVA DE LIMA

Aplicação de Mineração de dados e Aprendizagem de Máquina na detecção de conflitos entre políticas

RIO BRANCO - ACRE

2020, v-1.1.2

EDKALLENN SILVA DE LIMA

Aplicação de Mineração de dados e Aprendizagem de Máquina na detecção de conflitos entre políticas

Proposta de Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal do Acre como Exame de Qualificação à obtenção do Grau de Mestre em Ciência da Computação. Área de concentração: Engenharia de Sistemas e Informação

UNIVERSIDADE FEDERAL DO ACRE – UFAC

Programa de Pós-Graduação em Ciência da Computação – PPgCC

Orientador: PROF[a] DR[a]. LAURA COSTA SARKIS

RIO BRANCO - ACRE

2020, v-1.1.2

EDKALLENN SILVA DE LIMA

Aplicação de Mineração de dados e Aprendizagem de Máquina na detecção de conflitos entre políticas/ EDKALLENN SILVA DE LIMA. – RIO BRANCO - ACRE, 2020, v-1.1.2-

85p. : il. (algumas color.) ; 30 cm.

Orientador: PROFª DRª. LAURA COSTA SARKIS

Prop. Dissertação (Mestrado) – UNIVERSIDADE FEDERAL DO ACRE – UFAC
Programa de Pós-Graduação em Ciência da Computação – PPgCC, 2020, v-1.1.2.

1. Políticas de controle de acesso. 2. Mineração de Dados. 2. Aprendizagem de máquina. I. DRA. LAURA COSTA SARKIS. II. UFAC - Universidade Federal do Acre. III. Programa de Pós-Graduação em Ciência da Computação – PPgCC. IV. Título

EDKALLENN SILVA DE LIMA

Aplicação de Mineração de dados e Aprendizagem de Máquina na detecção de conflitos entre políticas

Proposta de Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal do Acre como Exame de Qualificação à obtenção do Grau de Mestre em Ciência da Computação. Área de concentração: Engenharia de Sistemas e Informação

Aprovada em <MES> de 2020.

BANCA EXAMINADORA

PROF[a]. DR[a]. LAURA COSTA SARKIS - Orientador, UFAC

PROF[o] DR. MANOEL LIMEIRA DE LIMA JUNIOR ALMEIDA, UFAC

PROF[a]. DR[a]. CATARINA DE SOUZA COSTA, UFAC

PROF[a]. DR[a]. ANA BEATRIZ ALVAREZ MAMANI - Suplente, UFAC

RIO BRANCO - ACRE
2020

Dedicatória: Este trabalho não seria possível sem a educação que me foi concedida, os ensinamentos e a sabedoria oriundos de você, Lucimar do Rego Albuquerque de Lima, muito mais que uma mãe, uma inspiração. <IN MEMORIAN>.

Agradecimentos

À Deus pela saúde e pelas condições de viver uma vida plena de significados, sentimentos, emoções e sentido.

À família maravilhosa, em todos os sentidos, que tenho. Pai, mães (sim, mais de uma), irmãos, irmãs, avós e avôs que se foram (e deixaram saudades eternas), tios, tias, primos e primas, agregados, enfim, todos que fazem a loucura que é qualquer festa "só com os parentes mais próximos". Eu amo todos vocês.

Às mulheres da minha vida, minha filha, Ana Ester e minha esposa, Vanessa Lima, por me suportarem, claro, mas, principalmente por me amarem incondicionalmente.

Mais uma vez, à você, Lucimar, nossa querida Lúcia, por sua sabedoria e ensinamentos. Pelo seu sorriso que sempre vinha cheio de significados. Pela sua vida ter sido a nossa vida. Por ter se doado tanto. Por ter trabalhado tanto para construir esta família linda (que, claro, nunca será a mesma sem você, jamais). Pelas inúmeras vezes em que você me chamava de "meu filho" (se eu soubesse que a contagem era regressiva tinha aproveitado mais). Por tudo o que você representou nas nossas vidas, obrigado. Nenhuma palavra que existe ou que será inventada em qualquer língua tem significado suficiente para descrever o que você era em nossas vidas. Na minha vida. Mais uma vez, obrigado...

Nunca te esquecerei...

"Assim como casas são feitas de pedras, a ciência é feita de fatos. Mas uma pilha de pedras não é uma casa e uma coleção de fatos não é, necessariamente, ciência".

Jules Henri Poincaré, matemático, físico e filósofo da ciência francês

Resumo

A quantidade de informações disponíveis cresce a cada ano. Aumenta, junto com o volume de dados e informações, o interesse em tratar, analisar e descobrir conhecimento a partir desta avalanche de dados. A mineração de dados juntamente com o aprendizado de máquina são duas ferramentas-chave dentro deste processo de descoberta de conhecimento e utilização de todos esses dados e informações para propósitos úteis. Entretanto, antes que os dados possam ser analisados, eles precisam ser armazenados e os sistemas computacionais sofrem, também de forma crescente, permanentes ameaças à sua segurança. Neste contexto se inserem as políticas para sistemas computacionais que buscam garantir meios para proteção, confidencialidade e confiabilidade dos acessos dos usuários aos objetos dentro dos sistemas de uma organização. Em sistemas com múltiplos sujeitos, muitas ações e diversos objetos, eventualmente, ocorrerão conflitos entre políticas. Um conflito ocorre quando os objetivos de duas ou mais políticas não podem ser atendidos simultaneamente em determinado contexto. Este trabalho tem como objetivo propor que o problema da detecção de conflitos em políticas pode ser convertido em um problema de *data mining* (mineração de dados) resolvido pela tarefa da classificação além de modelar e sintetizar uma forma de detectar estes conflitos mediante o uso de diferentes algoritmos e técnicas da aprendizagem de máquina que consigam acurácias elevadas e forneçam modelos genéricos o suficiente para serem usados em outros contextos.

Palavras-chave: Controle de Acesso. Mineração de dados. Aprendizagem de máquina. Conflitos diretos. Conflitos indiretos. Detecção de conflitos.

Abstract

The amount of information available grows every year. It increases, along with the volume of data and information, the interest in treating, analyzing and discovering knowledge from this avalanche of data. Data mining associated with machine learning are two key tools within this process of discovering knowledge and using all that data and information for useful purposes. However, before data can be analyzed, it needs to be stored and computer systems are also increasingly threatened with permanent security threats. In this context, policies for computer systems are inserted as a way to guarantee protection, confidentiality and reliability of users' access to objects within an organization's systems. In systems with multiple subjects, many actions and different objects, eventually, conflicts between policies will occur. A conflict occurs when the objectives of two or more policies cannot be met simultaneously in a given context. This work aims to propose that the problem of detecting conflicts in policies can be converted into a problem of data mining (data mining) solved by the task of classification, in addition to modeling and synthesizing a way of detecting these conflicts through the use of different machine learning algorithms and techniques that achieve high accuracy and provide generic models to be used in other contexts.

Keywords: Access control. Data mining. Machine learning. Direct conflicts. Indirect conflicts. Conflict detection.

Lista de ilustrações

Lista de códigos

Lista de tabelas

Lista de abreviaturas e siglas

AM	*Aprendizado de Máquina*
ATM	*Air-Traffic Management*
BPNN	*Basic Probabilistic Neural Network*
CPNN	*Constructive Probabilistic Neural Network*
DAC	*Discretionary Access Control*
DARPA	*Defense Advanced Research Projects Agency*
DoS	*Denial of Service*
GPGU	*General Purpose Graphic Processor Unit*
GPU	*Graphic Processor Unit*
KDD	*Knowledege Discovery in Data Bases*
MAC	*Mandatory Access Control*
ML	*Machine Learning*
MLP	*MultiLayer Perceptron*
NFL	*No-Free Lucnch*
PMC	*Perceptron MultiCamadas*
RBAC	*Role-Based Access Control*
ReBAC	*Relationship-Based Access Control*
RNA	*Rede Neural Artificial*
RTLS	*Real Time Location Systems*
SMA	*Sistema Multi Agentes*
SVM	*Support Vector Machiner*

Sumário

1 Introdução

Neste capítulo introdutório serão descritos o problema da pesquisa, a justificativa, a hipótese do trabalho, os objetivos, as soluções propostas, os resultados esperados, as limitações da pesquisa e como este trabalho está organizado.

1.1 Contextualização

De acordo com Alecrim (2019), o volume de dados e informações cresce exponencialmente a cada ano, portanto, há uma frequente e ininterrupta demanda por mais infraestrutura de TI nas empresas, nos governos e mesmo nos usuários domésticos e, mais ainda, por um correto tratamento, destino e interpretação à imensidão de dados gerados por pessoas, empresas e governos.(MACHADO, 2014)

Em uma ampla variedade de campos, os dados estão sendo coletados e acumulados em um ritmo acelerado e há, assim, uma crescente demanda por análise adequada destes.(FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996; LIMA; PEREIRA, 2012). Neste contexto se insere a mineração de dados com suas técnicas para tratamento e extração de conhecimento desse volume crescente de dados.(SILVA; PERES; BOSCARIOLI, 2017; FERRARI; SILVA, 2017)

Este trabalho usa diversas tarefas da mineração de dados para modelar uma hipótese que possibilite detectar conflitos em políticas de controle de acesso. Para isso, diversos algoritmos de classificação serão explorados, descritos e utilizados com ênfase nas redes neurais e outras técnicas lineares de classificação.

As políticas de proteção, confidencialidade e confiabilidade da informação, como as de controle de acesso, sendo parte da área de segurança computacional, são uma das formas de garantir, mediante o estabelecimento de regras, padrões e normas a salvaguarda e a disponibilidade das informações dos sistemas.(SARKIS, 2017)(BUI; STOLLER; LE, 2019).

Este tema, cf. Ueda (2012, p.1) "é um tema de pesquisa importante dentro do contexto de segurança de sistemas, pois é um dos componentes fundamentais em qualquer sistema de computação".

Segundo Li e Tripunitara (2006), um aspecto muito relevante e muitas vezes tratado com pouca ênfase na construção de sistemas é a formulação, gerenciamento e manutenção de políticas de segurança da informação, principalmente as de controle de acesso.

Nas palavras de Ueda (2012, p.1),

> A definição dessas políticas é normalmente orientada por modelos que fornecem um conjunto de regras e mecanismos para o funcionamento seguro de uma representação abstrata de sistemas. Porém, a administração de tais políticas frequentemente *se torna um processo complexo*, pois deve garantir que elas sejam eficientes e que não comprometam o *desempenho* dos sistemas. *[Grifo do autor.]*

Uma política, como as de controle de acesso, descrevem qual ação um sujeito (em um sistema) pode fazer (*permissão*), não pode fazer (*proibição*) ou é obrigado a fazer (*obrigação*) sobre um objeto em um dado contexto (SARKIS, 2017).

De maneira semelhante são conceituadas as *normas* e os conjuntos de normas usados para lidar com a autonomia e a diversidade de interesses entre os diferentes agentes em um sistema multiagentes como o descrito e estudado em Silvestre (2017).

Essas normas que regulam as ações dos agentes são análogas às definições das políticas citadas anteriormente nesta seção 1.1 e são, cf. Silvestre (2017) e Sarkis, Silva e Braga (2016) fatores importantes para garantir a eficácia da segurança dos sistemas.

Em contextos reais, porém, muitas vezes as políticas de segurança (e *normas*) apresentam conflitos entre si. Estes surgem quando, por exemplo, duas políticas regulando o mesmo comportamento de determinado objeto em um sistema estão ativas, mas uma delas obriga (ou permite) a realização de determinado comportamento ou ação enquanto a outra proíbe o mesmo. (SARKIS, 2017)(SILVESTRE, 2017).

A detecção automatizada destes conflitos, com alta acurácia e custo computacional conveniente, é o problema de pesquisa deste trabalho. A descrição pormenorizada dos conflitos entre políticas encontra-se na subseção 2.2.4 deste trabalho.

1.1.1 Problema

O problema investigado neste trabalho consiste na *detecção de conflitos de forma automatizada usando técnicas de mineração de dados e aprendizagem de máquina* que apresentem acurácias superiores a 95% e que, ao analisar simultaneamente várias políticas ao se inserir uma nova instância não leve a um custo computacional exponencial.

1.1.2 Justificativa

Na detecção de conflitos em políticas, geralmente, cf. revisão da literatura (no Capítulo 2), usam-se abordagens como as de Sarkis (2017) baseadas em análise de ontologias entre os atributos que compõem uma política e regras de propagação destas políticas ou procedimentos como os descritos em Silvestre (2017) que utilizam lógica deôntica[1] para encontrar os conflitos. Estas duas abordagens definem tipos de conflitos que podem ocorrer

[1] A lógica *deôntica* é um tipo de lógica usada para analisar de modo formal as normas e as proposições que tratam dessas normas (SILVESTRE, 2017)

entre políticas computacionais. Para detectá-los, estas políticas, nestes trabalhos, foram analisadas em pares (e sem filtros para agrupamentos) e mesmo quando foram verificadas múltiplas normas ou políticas (SILVESTRE, 2017), toda a base precisa ser "consultada" ou "varrida", a cada nova instância de uma política inserida ou analisada no sistema (para que o conflito seja ou não detectado).

Para Shoham e Tennenholtz (1995) esta forma de analisar políticas em pares é um problema NP-completo, ou seja, ainda não foi provado que esta classe de problemas pode ser resolvida em tempo polinomial, sendo assim, são tratados como computacionalmente custosos a cada vez que uma instância nova de política é analisada (em tempo de execução, sendo normalmente exponencial). Consequentemente, com o crescimento orgânico, natural e temporal das políticas em um sistema computacional, a manutenção e gerenciamento das políticas será, eventualmente, computacionalmente oneroso.

Alternativamente, técnicas e algoritmos de aprendizagem de máquina juntamente com as de mineração de dados foram utilizadas com resultados promissores na detecção de conflitos, principalmente em Obaidat e Macchairolo (1994), Chen (2011), bem como em Christodoulou e Kontogeorgou (2008) e Jin, Cheu e Srinivasan (2002). Estes estudos abordam problemas variados como detecção de colisões em voos, segurança de acesso computacional, incidentes em rodovias e intrusão de sistemas — todos de alguma forma relacionados à conflitos entre normas, regras, políticas ou direção.

Neste contexto e tendo em vista que: (i) em grandes organizações as políticas de segurança, como as de controle de acesso, pela quantidade de objetos, modalidades, sujeitos e ações inerentes a essas instituições tendem a ter grande quantidade de informações que aumentam diariamente e constantemente nos sistemas computacionais (FUGINI; BELLETTINI, 2004); (ii) que pode ocorrer com a análise de políticas em pares, conforme descrito no trabalho de Shoham e Tennenholtz (1995), um problema NP-completo que onera o custo computacional; (iii) que pode-se otimizar conhecimento adquirido e já existente nas organizações (os *datasets* de políticas).

Propõe-se neste trabalho, aplicar a mineração de dados com técnicas de aprendizagem de máquina, como possibilidade de solução na detecção de conflitos entre políticas computacionais tanto em tempo de design, quanto em tempo de execução, buscando minimizar o custo computacional, ao se aproveitar da "história" temporal das políticas da organização, mediante o conhecimento adquirido, "treinado" e otimizado pelos algoritmos de aprendizagem de máquina.

1.1.3 Hipótese

Diante do contexto apresentado na seção anterior e também por Fugini e Bellettini (2004) tem-se *como hipótese deste trabalho*, que o problema de detectar conflitos entre

políticas *pode ser convertido e transformado* em uma tarefa de classificação da mineração de dados e que o uso de algoritmos de aprendizagem de máquina associados a técnicas de *data mining* para detectar estes conflitos configure um método que apresente precisão e acurácia superiores a 95%.

1.2 Objetivos

Nesta seção serão descritos o objetivo geral e os específicos deste trabalho.

1.2.1 Objetivo geral

O objetivo deste trabalho é propor que o problema da detecção de conflitos em políticas pode ser convertido em um problema de *data mining* (mineração de dados) resolvido pela tarefa da classificação além de modelar e sumariar uma forma de detectar estes conflitos mediante o uso de diferentes algoritmos e técnicas da aprendizagem de máquina que consigam acurácias elevadas.

1.2.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

- Estabelecer a relação entre machine learning, técnicas de mineração de dados e o problema do conflito entre políticas;
- Determinar e comparar quais algoritmos e técnicas são mais adequados para cada tipo de conflito nas políticas (ou normas) usando as suas acurácias;
- Usar e comparar o desempenho, a precisão e taxa de acertos das principais técnicas usadas no aprendizado de máquina: redes neurais artificiais (RNA), Support Vector Machines (SVM) e redes neurais recorrentes e profundas.
- Transformar o problema da detecção de conflitos em uma tarefa de classificação da mineração de dados mantendo acurácias elevadas;
- Usar *frameworks* de aprendizado de máquina como TensorFlow (KADIMISETTY, 2018) ou Torch (PASZKE et al., 2019), na construção, treinamento e teste de redes neurais, comparando-os, quando adequado, para o problema específico deste trabalho;
- Estabelecer a detecção de conflitos em políticas (ou normas) como uma classe de problemas a serem resolvidos de forma eficiente por técnicas de aprendizagem de máquina.

1.3 Solução Propostas

Diante da hipótese apresentada na subseção 1.1.3, a solução para o problema apresentado neste trabalho na subseção 1.1.1 concentra-se, prioritariamente, em mostrar que *converter* (ou *transformar*) a detecção de conflitos a um *problema de classificação* da mineração de dados associado a técnicas de aprendizagem de máquina, reestruturando os atributos do *dataset*, se necessário, se configura um método com acurácia superior a 95%.

A primeira solução proposta para conflitos diretos entre políticas é usar as técnicas e algoritmos de classificação (aprendizado supervisionado) para realizar a detecção automatizada de conflitos. Para isso, propõem-se:

- Usar, inicialmente, uma rede neural (um perceptron de uma camada ou com somente uma camada oculta e apenas com *forward*) como técnica algorítmica para a detecção de conflitos e
- Construir a arquitetura de uma rede neural multicamadas (com camadas ocultas), e retropropagação (*backpropagation*), comparando-a com outro classificador linear, como, por exemplo, o SVM, para estabelecer qual técnica de mineração de dados na detecção de conflitos em políticas é mais precisa.

Realizado os múltiplos experimentos, atestar a hipótese mediante os resultados apresentados.

1.4 Método de Pesquisa

O método de pesquisa relatando todos os passos necessários para demonstrar os objetivos descritos na seção 1.2 e de que forma eles foram atingidos serão pormenorizadamente detalhados no Capítulo 3 deste trabalho.

1.5 Resultados Esperados

Ao fim deste trabalho os seguintes resultados são esperados:

- Mostrar que o problema da detecção de conflitos em políticas pode ser convertido em um problema de *data mining* (mineração de dados) resolvido pela tarefa da classificação;
- Demonstrar que o problema da detecção de conflitos é um prolema linearmente separável;

- Mostrar que a política nova (*instância inédita*) é ou não conflitante imediatamente após a criação da mesma usando como base o treinamento da rede neural no *dataset* de políticas existente;
- Modelar e resumir uma forma de detectar estes conflitos mediante o uso de diferentes algoritmos e técnicas da aprendizagem de máquina que consigam acurácias superiores a 95%;
- Estabelecer a relação entre machine learning, técnicas de mineração de dados e o problema do conflito entre políticas;
- Determinar e comparar quais algoritmos e técnicas são mais adequados para cada tipo de conflito nas políticas (ou normas) usando as suas acurácias;
- Usar e comparar o desempenho, a precisão e taxa de acertos das principais técnicas usadas no aprendizado de máquina: redes neurais artificiais (RNA), Support Vector Machines (SVM); redes neurais recorrentes e profundas;
- Estabelecer a detecção de conflitos em políticas (ou normas) como uma classe de problemas a serem resolvidos de forma eficiente por técnicas de aprendizagem de máquina.

1.6 Limitações do Trabalho

Não faz parte do escopo deste trabalho:

- Delinear um modelo de política com objetivos semânticos diferenciados. Para os experimentos deste trabalho será usado o modelo de políticas descrito em Sarkis (2017) e em Sarkis, Silva e Braga (2016)
- Analisar comparativamente os modelos de extensão de políticas em um determinado contexto;
- Usar redes neurais convolucionais profundas na detecção dos conflitos;
- Abordar a semântica em políticas;

1.7 Organização do trabalho

Este trabalho está organizado da seguinte forma:

- No Capítulo 2 apresenta-se todo o referencial teórico compreendendo uma revisão bibliográfica sobre os principais temas desta proposta de dissertação, como políticas,

detecção de conflitos, mineração de dados e aprendizagem de máquina (e seus algoritmos principais)

- No Capítulo 3 são mostrados o método e os experimentos e resultados obtidos.
- No Capítulo 4 mostra-se o cronograma para a finalização da dissertação
- No Capítulo 5 apresenta-se as conclusões atingidas e esperadas desta pesquisa.

2 Revisão Bibliográfica

Neste capítulo, alguns conceitos fundamentais da pesquisa bibliográfica realizada serão explanados com o intuito de atingir os objetivos descritos na seção 1.2 e o entendimento da solução proposta neste trabalho. Serão abordados modelos de políticas, como as de controle de acesso e conflitos entre as mesmas. Temas como mineração de dados, aprendizagem de máquina, algoritmos de classificação, com destaque para as **RNA's** –*Redes Neurais Artificiais* e **SVM** — *Support Vector Machines*, que são a base da hipótese deste trabalho também serão descritos e suas definições teóricas serão discutidas. Ao final, serão descritos alguns trabalhos relacionados ao tema desta proposta de dissertação.

2.1 Segurança de dados computacionais

As informações e os dados disponíveis sofreram um aumento frenético nas últimas décadas, fato que gerou uma crescente preocupação com o correto tratamento destes dados e, principalmente com questões de segurança relacionadas à proteção destas informações que seguem, a cada ano sendo coletadas e acumuladas em ritmo vertiginoso (ALECRIM, 2019; MACHADO, 2014; LIMA; PEREIRA, 2012; FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996; SANTOS, 2007).

Considerando, por exemplo, o ambiente corporativo, cf. Fontes (2012, p.1), a "informação é um recurso essencial para toda organização, independentemente do seu porte e do seu segmento de atuação no mercado".

Neste contexto, tanto a obtenção da informação é um processo importante quanto o são as formas de armazenamento e proteção, além, claro, o fato de que a análise e interpretação destes dados e informações tornam-se atividades essenciais para a manutenção de negócios e o próprio desenvolvimento da sociedade. (SILVA; PERES; BOSCARIOLI, 2017) (MARCIANO, 2006).

A informação, como observa-se, é um recurso crítico para qualquer instituição exigindo, portanto, *a adoção de políticas de segurança* adequadas que visem a sua proteção, salvaguarda e manutenção para que este ativo, tão significativo, mantenha seu valor, sua abrangência e importância dentro do cenário das organizações, governos e usuários comuns (MARCIANO, 2006).

Isto porque assim como a quantidade de dados e informações cresce rapidamente, aumenta também, a cada ano, *as ameaças* aos ativos de informação variando desde fraudes informáticas e sabotagens até vandalismo ou espionagem (CASACA; CORREIA, 2013).

Nos primórdios da computação o mundo ainda não estava totalmente interconectado

por redes de computadores, em especial, pela Internet que trouxe, além de inúmeros benefícios, diversos outros problemas de segurança da informação e representou um desafio novo à proteção dos dados das organizações, governos e usuários domésticos (FONTES, 2012).

Assim, para Mukkamala, Janoski e Sung (2002), observa-se que que a segurança da informação é uma questão de preocupação global séria. Pois, a complexidade, acessibilidade e capilaridade da Internet serviram para aumentar as ameças à segurança dos sistemas de informação.

Fontes (2012, p. 2) afirma que, por exemplo, informações que antes estavam persistidas em relatórios e poderiam ser protegidas fisicamente em gavetas ou armários hoje podem ficar disponíveis na Internet e acessíveis no mundo todo. Mínimas falhas ou ações criminosas podem disponibilizar informações sigilosas ou privadas ou bloquear o acesso a informações críticas para a realização do objetivos da organização, como alguns ataques cibernéticos o fazem.

Portanto, para Casaca e Correia (2013), estes ativos de informação *devem ser protegidos* de ameaças através de estratégias e *políticas de segurança da informação*, feitas com base em modelos e métricas bem conhecidas e previamente definidas, permitindo às organizações melhorar seu processo de segurança da informação.

Assim, cf. Doherty e Fulford (2005), como as informações são um ativo corporativo crítico que se tornou cada vez mais vulnerável a ataques de vírus, hackers, criminosos e erros humanos, as organizações precisam priorizar a segurança de seus sistemas para garantir que seus ativos de informação mantenham sua precisão, confidencialidade e disponibilidade.

2.2 Controle de acesso

Para Knapp et al. (2009), o objetivo de uma política é fornecer orientação gerencial e suporte à segurança da informação de acordo com os requisitos de negócios e as leis e regulamentos relevantes em determinada organização ou instituição. A política de segurança em um sistema computacional garante, portanto, a proteção de suas informações. Dentre as diversas tecnologias utilizadas para assegurar essas propriedades, temos, por exemplo, o controle de acesso (SARKIS; SILVA; BRAGA, 2016)

Para Wang et al. (2010), o controle de acesso é o mecanismo central para atingir os requisitos de segurança em sistemas de informação. Dessa forma, trata-se de uma tecnologia indispensável para quem faz uso de qualquer tipo de sistema, podendo basear-se ou coexistir com outros serviços de segurança (SANDHU; SAMARATI, 1996).

Kropiwiec (2005, p. 8) afirma que "para entender o que é e como funcionam os mecanismos de controle de acesso de um sistema [...], faz-se necessário classificar os

elementos do sistema em três grupos." Que são, segundo este autor, o conjunto de *sujeitos*, de *objetos* e o de *ações*.

De acordo com Kropiwiec (2005, p. 8), o *conjunto de sujeitos* "englobam qualquer elemento que pode realizar ações sobre objetos, e é composto por usuários, processos e o próprio sistema". Já o *conjunto de objetos* "englobam os elementos sobre os quais podem ser realizadas ações". E, por sua vez, o *conjunto de ações* "compreende a lista de ações que podem ser realizadas sobre cada um dos objetos do sistema". Como, por exemplo, ler, escrever, apagar, abrir conexão, enviar mensagem, encerrar conexão entre muitas outras ações relativas a cada sistema em particular.

Os mecanismos de controle de acesso, portanto, verificam todas as requisições de dados e recursos administrados pelos sistemas definindo, assim, as circunstâncias nas quais as requisições de acesso são permitidas ou negadas (SANDHU; SAMARATI, 1996).

Desta forma, tanto as informações quanto os ativos do sistema mantém um nível adequado de segurança, conservam o sigilo e estabelecem importantes providências quanto ao acesso a informações ou recursos confidenciais ou protegidos (WANG et al., 2010).

Com o controle de acesso o usuário fica limitado apenas a execução de operações e ações no sistema que lhe foram previamente concedidas. Ao requerer acesso ao conteúdo de informações ou permissão de uso de recursos estes só serão outorgados a quem possui o direito de acesso aos mesmos (FERRAIOLO et al., 2001).

Os modelos de controle de acesso fornecem um conjunto de regras e mecanismos para o funcionamento seguro dos sistemas, sendo responsáveis pela definição de políticas específicas de controle de acesso. Para Monteiro (2017), as políticas são diretrizes de *alto nível* que determinam como os acessos são controlados e decisões de acessos são estabelecidas (VIMERCATI; SAMARATI; JAJODIA, 2005) (SARKIS, 2017) (LOPES, 2012).

Embora não seja o escopo deste trabalho é importante salientar que existem vários modelos, na literatura, de controle de acesso. Entre os principais, pode-se citar, o controle de acesso discricionário (*Discretionary Acces Control* - DAC); o controle de acesso mandatório (*Mandatory Access Control* - MAC); o controle e acesso baseado em papéis (*Role-Based Access Control* - RBAC) e o controle de acesso baseado em relacionamentos (*Relationship-Based Access Control* - ReBAC). Sobre o DAC e o MAC, sugere-se os trabalhos de Sandhu e Samarati (1996) e Vimercati, Samarati e Jajodia (2005). Sobre RBAC, recomenda-se Ferraiolo et al. (2001) e Sandhu e Samarati (1996). Sobre o ReBAC aconselha-se seguir o trabalho de Bui, Stoller e Le (2019) e Fong (2011).

2.2.1 Políticas de Controle de acesso

Uma política de controle de acesso tem como objetivo definir ou limitar o comportamento atual ou futuro de *sujeitos* e *objetos* para garantir que as suas *ações* estejam alinhadas com os objetivos da empresa de acordo com o escopo de acesso de cada sujeito ou grupo de sujeitos, com as permissões, proibições ou obrigações que estes tenham sobre os objetos e quais dados ou recursos lhe são concedidos (DUNLOP; INDULSKA; RAYMOND, 2002)(SARKIS, 2017).

As políticas de controle de acesso convencionais, inicialmente foram chamadas de *autorizações* e tinham a seguinte forma: `{sujeito, objeto, ação}`. Estas *autorizações* especificavam quais operações os *sujeitos* podiam executar sobre os *objetos* no contexto de um sistema (VIMERCATI; SAMARATI; JAJODIA, 2005) (SARKIS, 2017).

Com o desenvolvimento dos sistemas e das políticas, estas últimas passaram a ser direcionadas, principalmente, na especificação e administração de requisitos de controle de acesso expressos na forma de *proibições*, *permissões* e, posteriormente de *obrigações* que são as composições principais na aplicação destas políticas (SARKAR; BALI; SHARMA, 2017).

2.2.2 Modelos de políticas

Os autores Moffett e Sloman (1994) afirmam que um modelo de política deve ter os seguintes atributos fundamentais: `{modalidade, sujeito, objeto e ação}`. A particularidade da política compreende estabelecer uma *autorização*, uma *permissão* ou *proibição*.

Para Moffett e Sloman (1994), o *sujeito* da política é a quem ela é orientada. O *objeto* define o conjunto de objetos no qual a política está focada. A *ação* é estabelecida como procedimentos que podem ser efetuados em *objetos* no sistema. Outros trabalhos da literatura também consideram estes atributos apresentados aqui, com as mesmas conotações, para a definição de uma política, como o trabalho de Sarkis (2017).

Outros modelos de políticas, cada um com as suas particularidades, são descritos em Lupu e Sloman (1999), também em Sloman e Lupu (2002) e Koch, Mancini e Parisi-Presicce (2002) além de Bui, Stoller e Le (2019) e Dunlop, Indulska e Raymond (2002), mas não serão utilizados neste trabalho.

2.2.3 Modelo de Política utilizado

De acordo com Sarkis (2017, p. 36):

> Definir uma política de controle de acesso não é uma tarefa simples, principalmente porque algumas vezes é necessário representar formalmente

> políticas complexas, tais como as que tem origem em práticas de leis e regulamentos organizacionais.

Desta forma, a definição da política deve combinar todos os diferentes regulamentos para ser executada e considerar todas as possíveis ameaças adicionais relativas ao uso de sistemas (VIMERCATI; SAMARATI; JAJODIA, 2005).

O modelo de política utilizado neste trabalho baseia-se inteiramente no modelo proposto por Sarkis (2017) e Sarkis, Silva e Braga (2016) baseado nos modelos, ele próprio, nos propostos pelos autores Cuppens, Cuppens-Boulahia e Ghorbel (2007), e também o trabalho de Elrakaiby, Cuppens e Cuppens-Boulahia (2012), bem como no artigo de Kalam et al. (2003).

Portanto, de acordo com Sarkis (2017, p.36) e Sarkis, Silva e Braga (2016):

> Uma política é uma tupla da forma:
>
> $$Policy = KP \times Org \times SR \times AA \times OV \times Ac \times Dc \tag{2.1}$$
>
> Onde **KP** descreve o tipo de política (uma proibição (F), da palavra em inglês Forbidden; uma permissão (P); ou uma obrigação (O)). **Org**. relata o local (ambiente) onde a política deve ser cumprida, isto é, a organização na qual os sujeitos devem cumprir a política. **SR** descreve a quem (entidades) se destina a política (pode ser um sujeito $s \in S$ ou um papel $r \in R$, ou seja, $SR = S \cup R$. Um sujeito pode ser um usuário $u \in U$ ou uma organização $org \in Org$ representando o grupo de sujeitos que devem cumprir com a política, isto é, $S = U \cup Org$). **AA** identifica uma ação $a \in A$ ou uma atividade $act \in Act$ (uma atividade é a união de várias ações relacionadas). **OV** relata um objeto $o \in O$ ou uma visão $v \in V$ que está sendo manipulada pela ação/atividade (uma visão é a união de vários objetos). **Ac** é a condição de ativação da política e **Dc** é a condição de desativação da política. Uma condição constitui a configuração para um evento, em termos de que a política deva seguir.

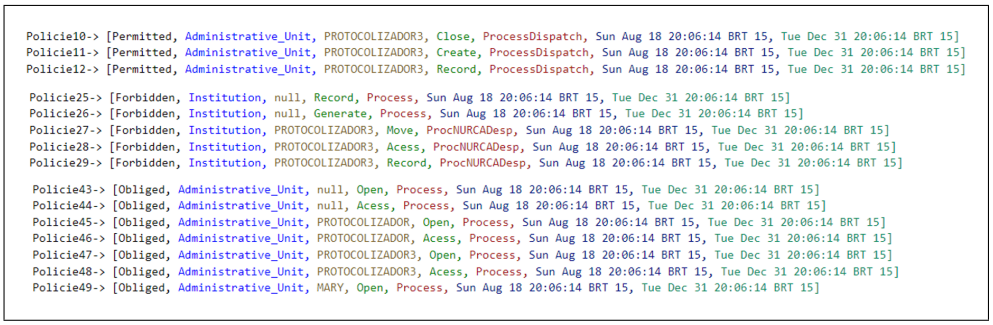
Em Sarkis (2017) foi definido Ac e Dc como datas, desta forma Ac é a data de ativação da política e Dc é a data de desativação.

A Figura 1 exemplifica o modelo de políticas utilizadas neste estudo para a mineração de dados e aprendizagem de máquina.

2.2.4 Detecção de conflitos

Segundo Kalam et al. (2003), quando um modelo de controle de acesso inclui a possibilidade de especificar *permissões*, *proibições* e *obrigações* podem ocorrer alguns conflitos entre políticas.

Figura 1 – Modelo das políticas utilizadas no estudo



Fonte: compilação do autor

Os conflitos podem acontecer quando diferentes conjuntos de condições resultam em *permitir* e *negar* simultaneamente, ao mesmo papel, à mesma solicitação, ou *proibir* e *obrigar* o mesmo papel, à mesma solicitação, isto é, quando os *objetivos* de duas ou mais políticas *não podem ser atendidos simultaneamente* (CUPPENS; CUPPENS-BOULAHIA; GHORBEL, 2007).

2.2.4.1 Classificação dos conflitos

De acordo com Cuppens, Cuppens-Boulahia e Ghorbel (2007), Sloman e Lupu (2002) e Lupu e Sloman (1999), os conflitos classificam-se, principalmente em:

- *conflito de modalidade* que decorre de políticas de propriedades contrárias;
- *conflito em potencial* surge da tripla sobreposição de sujeitos, ações e objetos de políticas de predicados opostos;
- *conflito de redundância* que surge da precedência de execução dadas a certas políticas e
- *conflito específico de aplicação* que ocorre quando ações antagônicas são atribuídas para a mesma entidade (sujeito ou papel) mediante controles externos específicos, expressos como metapolíticas que agem como contenções para políticas permitidas.

Neste trabalho, para o modelo de política proposto, não abordam-se os conflitos de redundância já que ordens de precedência não são tratadas. Todas as outras modalidades de conflitos são examinadas e utilizadas.

Assim como em Sarkis (2017, p. 24), neste trabalho, os conflitos em potencial são denominados **conflitos diretos** e os conflitos de aplicação são abarcados pelos chamados **conflitos indiretos**.

2.2.4.2 Conflitos Diretos e Indiretos

Dunlop, Indulska e Raymond (2002) usam uma abordagem para os conflitos diretos e simples que será replicada neste trabalho. Diz-se que duas regras estão em conflito *quando o cumprimento de uma das regras viola a outra e vice-versa.* Ou seja, a verificação se há o conflito é feita entre duas políticas que possuem modalidades contraditórias ou antagônicas, definidas na mesma organização, executadas pelos mesmos sujeitos, efetuando a mesma ação em relação a um objeto específico.

Exemplo de um ***conflito direto***:

`{P1= Permitido, na Universidade X, Ana Ester, acessar processos administrativos}`

`{P2= Proibido, na Universidade X, Ana Ester, acessar processos administrativos}`

Os exemplos acima mostram que quando uma política proíbe e a outra permite um sujeito de realizar uma ação estabelecida sobre um objeto específico em uma organização particular ocorre um conflito direto. O conflito, segundo Autrel, Coma e al (2008), pode ser identificado diretamente utilizando a sobreposição dos atributos das políticas.

Já em um conflito indireto, as políticas conflitantes regulam ações diferentes (mas relacionadas) executadas por distintos sujeitos (porém, relacionados) sobre objetos desiguais (mas, relacionados) em organizações diferentes (mas, relacionadas) (SARKIS, 2017, p.24).

Além disso, um conflito indireto pode ainda ocorrer, mesmo quando as políticas em conflito não têm modalidades contraditórias ou contrárias.

Ex:

`P3 = Obrigado, Empresa E, Funcionário, receber, avaliação, mensal`

`P4= Permitido, Empresa E, Analista, conceder, avaliação, mensal`

Este conflito não seria detectado diretamente, porém há um conflito se considerarmos os relacionamentos.

A capacidade de um sistema reconhecer um estado inconsistente em andamento ou em potencial é denominada **detecção de conflitos.**

Para Sarkis (2017),

> detectar conflitos entre políticas de controle de acesso é o primeiro passo para buscar inibir o surgimento de erros no sistema relativo às políticas aplicadas, tendo em vista que as políticas sem conflitos refletem corretamente o plano de segurança do sistema.

Ainda segundo Sarkis (2017, p.25), "Para um sistema baseado em políticas trabalhar de forma eficaz é importante ter um meio de detectar e resolver os conflitos que possam surgir". Torna-se, assim, indispensável a utilização de abordagens que analisem previamente a existência de conflitos como o proposto nesta proposta de dissertação.

Na próxima seção os conceitos de mineração de dados utilizado neste trabalho serão descritos.

2.3 Mineração de Dados

Uma das características de nossa era é produção de dados em grande volume, velocidade e variedade de todas as formas, por dispositivos espalhados em toda parte. Entretanto, dados, mesmo em grande quantidade, são apenas dados. É preciso produzir informação e conhecimento para explorar as vantagens que essa massa pode trazer. O dado necessita ser, de alguma forma, analisado, tratado para que informações e conhecimento possam ser, deles, extraídos (AMARAL, 2016) (FERRARI; SILVA, 2017).

Conforme Fayyad, Piatetsky-Shapiro e Smyth (1996):

> Os computadores permitiram que os humanos coletassem mais dados do que podemos digerir, é natural [,portanto,] recorrer a técnicas computacionais para nos ajudar a desenterrar padrões e estruturas significativas a partir dos numerosos volumes de dados. Por isso, [a mineração de dados] é uma tentativa de resolver um problema que a era da informação digital transformou em realidade para todos nós: sobrecarga de dados.

Para Silva, Peres e Boscarioli (2017), a *mineração de dados* pode ser definida como um processo automatizado ou semiautomatizado de explorar grandes bases de dados de forma extensiva, com o objetivo de encontrar padrões relevantes que ocorrem nos dados e que sejam significativos para embasar a absorção de informação importante, contribuindo para a geração de conhecimento.

Para Fayyad, Piatetsky-Shapiro e Smyth (1996), o termo "mineração de dados" tem sido usado por estatísticos, analistas de dados e comunidades de sistemas de informações ganhando popularidade no campo do banco de dados. Já o termo *descoberta de conhecimento em bancos de dados* [(KDD, da sigla em Inglês)] foi cunhada para enfatizar que o conhecimento é o produto final de uma descoberta baseada em dados. Está sendo utilizado nos campos de IA e aprendizado de máquina.

2.3.1 KDD - Knowledge Discovery in Databases

Assim, a mineração de dados é parte integrante de um processo mais amplo, conhecido como descoberta de conhecimento em bases de dados (*Knowledge Discovery in Databases*, ou *KDD*)(FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996).

Embora se use *mineração de dados* como sinônimo de KDD, a terminologia é empregada para a etapa de *descoberta* do processo de KDD, que inclui a *seleção* e *integração* das bases de dados, a *limpeza* da base, a *seleção e transformação* dos dados,

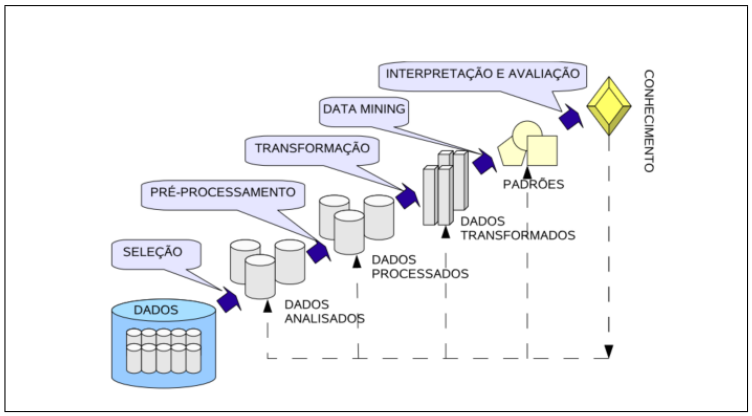
a *mineração*(propriamente) e a *avaliação* dos dados (FERRARI; SILVA, 2017)(SILVA; PERES; BOSCARIOLI, 2017).

Assim, a mineração de dados é definida em termos de esforços para a descoberta de padrões em bases de dados. A partir destes padrões descobertos, há condições de se gerar conhecimento útil para um processo de tomada de decisão (ou a geração de conhecimento para esta tomada).

Mais especificamente, KDD (*Knowledge Discovery in Database*) é um processo de busca de conhecimento em bancos de dados e, de modo geral, consiste de uma sequência iterativa de passos (ou **etapas**)[1]: limpeza de dados; integração dos dados; seleção, transformação e mineração dos dados; avaliação dos padrões e apresentação e assimilação do conhecimento. Este processo é iterativo e, em alguma etapa, pode-se voltar para uma anterior (SILVA; PERES; BOSCARIOLI, 2017).

A Figura 2 mostra o funcionamento iterativo do processo de KDD (*Knowledge Discovery in Database*) - Descoberta de Conhecimento em Bases de Dados.

Figura 2 – Etapas do processo de descoberta do conhecimento em bases de dados - KDD



Fonte: Vasconcelos e Carvalho (2018, p. 7)

Neste trabalho as tarefas de seleção e transformação dos dados farão parte da etapa chamada de pré-processamento de acordo com, Silva, Peres e Boscarioli (2017) e serão descritas com maior riqueza de detalhes no Capítulo 3.

2.3.2 Modelo de conhecimento

O termo **modelo de conhecimento** (ou hipótese) é utilizado na literatura (e neste trabalho) para fazer referência a um padrão ou conjunto de padrões descobertos (que é, enfim, o *propósito* do processo de KDD). Estes padrões são conhecimentos representados segundo as normas sintáticas de alguma linguagem formal. Estes padrões podem ser classificados em dois tipos: *preditivos* e *descritivos* (FERRARI; SILVA, 2017).

[1] O processo de KDD, segundo (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996) é composto por: *Seleção de dados; Pré-processamento; Transformação; Mineração; Análise e assimilação de resultados*

O intuito dos preditivos é resolver um problema específico de prever os resultados ou valores de um ou mais atributos, em função dos valores de outros atributos. Os descritivos (ou informativos) tem o intuito de apresentar informações interessantes e importantes sobre os dados que um especialista de domínio possa não conhecer (Goldschmidt; Passos, 2005).

Modelos de conhecimento compostos exclusivamente por padrões preditivos são chamados de modelos preditivos, enquanto que modelos descritivos são modelos de conhecimento compostos por padrões descritivos (SILVA; PERES; BOSCARIOLI, 2017).

Para Minewiskan (2020),

> Um modelo de mineração é criado aplicando-se um algoritmo a dados, mas é mais que um algoritmo ou um contêiner de metadados: é um conjunto de dados, estatísticas e padrões que podem ser aplicados a novos dados para gerar previsões e fazer inferências sobre relações.

Neste contexto, este trabalho se concentra, portanto, em criar modelos de forma a detectar, mediante o uso de técnicas da mineração de dados (e aprendizagem de máquina) os conflitos entre as políticas de controle de acesso de um sistema. Diversos modelos serão desenvolvidos e confrontados usando-se vários algoritmos e técnicas analisados com métricas específicas.

2.3.2.1 Arquitetura do modelo

Segundo Minewiskan (2020), "Um modelo de mineração obtém dados de uma estrutura de mineração e analisa esses dados usando um algoritmo de mineração de dados". Entretanto, é importante diferenciar a estrutura e o modelo de mineração. Ainda de acordo com Minewiskan (2020), a *estrutura* armazena informações que definem a fonte de dados, já um *modelo* de mineração armazena informações derivadas do processamento estatístico dos dados, como padrões encontrados em decorrência da investigação.

Assim, o modelo fica "limpo" até que os dados que foram guarnecidos pela estrutura de mineração sejam processados e avaliados. Depois de produzido o modelo contém *metadados*, resultados e associações e pode, então, ser utilizado para a obtenção de conhecimento.

A arquitetura pode conter também, variáveis, *hiperparâmetros*, definições do modelo, filtros utilizados e, claro, o algoritmo utilizado na tarefa de análise dos dados (ACADEMY, 2020).

2.4 Aprendizagem de máquina

Segundo Goldschmidt e Passos (2005, p. 10),

> um dos passos do processo de KDD, o de extração de padrões (ou Mineração de Dados) utiliza métodos de Aprendizado de Máquina (AM) [ou *Machine Learning* - ML] para encontrar regularidades, padrões ou conceitos em conjuntos de dados.

A principal diferença, segundo os autores, Goldschmidt e Passos (2005) entre Aprendizagem de Máquina e KDD reside no fato de "grande parte da literatura em AM se concentra apenas no mecanismo de descoberta de padrões e/ou conceitos, sem se preocupar com o grau de utilidade". Já em KDD, ainda segundo Goldschmidt e Passos (2005), "os padrões extraídos são avaliados para aferir sua utilidade para o usuário em relação à tomada de decisão". Ou seja, o aprendizado na Aprendizagem de Máquina é um atividade-fim enquanto o aprendizado no KDD é uma atividade-meio para a obtenção do conhecimento.

2.4.1 Definição

Aprendizado de máquina ou *machine learning* é um braço da Inteligência Artificial que emprega técnicas e algoritmos na criação de modelos computacionais dos quais a característica principal é a capacidade de descobrir padrões em um grande volume de dados ou de melhorar o desempenho de uma determinada tarefa através da experiência (do *reforço*) (MOHRI; ROSTAMIZADEH; TALWALKAR, 2018) (ALPAYDIN, 2014) (SWAMYNATHAN, 2019)

Então, de acordo com Baeza-Yates e Ribeiro-Neto (2013, p. 278), "os padrões aprendidos, que podem ser bem complexos, são então usados para fazer predições relativas a dados ainda não vistos e novos".

Nas palavras de Arthur Lee Samuel Wiederhold e McCarthy (1992), considerado um dos pioneiros na área de inteligência artificial, aprendizado de máquina é "o campo de estudo que dá aos computadores a capacidade de aprender sem serem explicitamente programado" (SIMON, 2013, p. 89).

Aprendizado de máquina tem sido aplicado na automatização de funções que para os humanos são executadas intuitivamente, mas que são difíceis de definir formalmente (SARKAR; BALI; SHARMA, 2017).

Assim, de forma geral, a aprendizagem de máquina tem por objetivo estudar e desenvolver métodos computacionais para obter sistemas capazes de adquirir conhecimento de forma automatizada (LIMA; PINHEIRO; SANTOS, 2016).

A capacidade de determinados algoritmos tem de aprender a partir de exemplos é chamado de **aprendizado indutivo**. Estes algoritmos aprendem relacionamentos eventualmente existentes entre os dados, mostrando o resultado nos modelos de conhecimento gerado (Goldschmidt; Passos, 2005) (ALPAYDIN, 2014).

Conforme Baeza-Yates e Ribeiro-Neto (2013, p. 279), os algoritmos de aprendizado de máquina "são fundamentalmente dependentes de uma fase de aprendizado, a qual é usada para produzir um modelo ou uma função que codifica padrões presentes nos dados de entrada".

Então, dependendo de qual é a abordagem de aprendizado usada, os algoritmos de aprendizado de máquina podem ser, basicamente de 3 principais tipos, que são: a aprendizagem supervisionada, a aprendizagem não-supervisionada e a aprendizagem por reforço [2] (RUSSELL; NORVIG, 2013) (BAEZA-YATES; RIBEIRO-NETO, 2013).

Na **aprendizagem não-supervisionada** o modelo/hipótese busca padrões na entrada, embora não seja fornecido nenhum *feedback* explícito. Portanto, na abordagem não-supervisionada não há, nos dados, uma classe, não há um rótulo prévio, ou seja, não existe a informação da saída desejada. O processo de aprendizado busca identificar regularidades entre os dados e não é necessária a divisão prévia dos dados em dados de treinamento, validação e teste. A tarefa mais comum de aprendizagem não supervisionada é o agrupamento. mas, os algoritmos de aprendizagem não-supervisionada incluem ainda, modelos de redes neurais, análise de componentes independentes e o já citado *clustering* (agrupamento) (RUSSELL; NORVIG, 2013) (SILVA; PERES; BOSCARIOLI, 2017) (Goldschmidt; Passos, 2005) (AMARAL, 2016).

Na **aprendizagem supervisionada** o modelo/hipótese observa alguns exemplos de pares de entrada e saída, e aprende uma função (ou modelo) que faz o mapeamento entre a entrada e a saída. Portanto, ela compreende a abstração de um modelo a partir dos dados apresentados na forma de pares ordenados (*entrada, saída, saída desejada*). Há, assim, uma *classe*, ou um atributo especial com o qual se pode comparar e validar o resultado. Esta categoria de aprendizagem de máquina requer uma função de aprendizado dos dados de treinamento fornecidos como entrada. Esses dados, então, são usados para para aprender uma função de classificação que pode, assim, ser usada para realizar predições de classes para dados ainda não vistos ou novos (RUSSELL; NORVIG, 2013) (LUGER, 2015) (BAEZA-YATES; RIBEIRO-NETO, 2013).

Na **aprendizagem por reforço**, aprende-se a partir de uma série de reforços — recompensas ou punições. Não está disponível, geralmente, na aprendizagem por reforço, para o algoritmo de aprendizado de máquina, um conjunto de dados para treinamento. O aprendizado se dá, então, pela interação com o ambiente que se deseja atuar por um determinado período com o objetivo de melhorar o desempenho de uma determinada tarefa (RUSSELL; NORVIG, 2013) (AMARAL, 2016) (SILVA, 2019).

[2] Há ainda os tipos de aprendizado *semissupervisionado* e a *transdução* (ou inferência transdutiva) que não serão discutidos neste trabalho

2.5 Algoritmos de classificação

A classificação é considerada uma das tarefas principais do processo de aprendizagem de máquina, sendo, inclusive, a tarefa mais comum (AMARAL, 2018) (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996).

Segundo, Amaral (2018, p. 86), "Na classificação, os dados devem possuir uma *classe* a qual queremos prever". Conforme Rocha et al. (2012), "o termo *classe* deve ser usado quando existe informação sobre quantas e quais são as partições presentes em um conjunto de dados, bem como qual exemplar pertence a qual partição".

Comumente, denomina-se *classificação* o processo pelo qual se determina uma função de mapeamento capaz de indicar a qual classe pertence algum exemplar de um domínio sob análise, baseando-se em um conjunto já classificado (SILVA; PERES; BOSCARIOLI, 2017).

Assim, de acordo com Kesavaraj e Sukumaran (2013), classificação é uma técnica de mineração de dados (aprendizado de máquina) usada para prever a associação ao grupo para instâncias de dados. É, segundo Amaral (2016) e Shazmeen, Mustafa e Baig (2013), como citado anteriormente, a tarefa mais utilizada em mineração de dados. Além de ser a mais complexa e a que possui a maior quantidade de algoritmos disponíveis, conforme descrito em Kesavaraj e Sukumaran (2013).

A classificação é uma das tarefas *preditivas* de Mineração de Dados e aprendizado de máquina. Tarefas de predição consistem na análise de um *dataset* (conjunto de dados), descritos por atributos e rótulos associados com o objetivo de descobrir um **modelo** capaz de mapear corretamente cada um dos dados a seus rótulos apropriadamente. Esse objetivo é alcançado por meio de técnicas, normmalmente, chamadas de supervisionadas. Este tipo de análise preditiva pode ser dividida em *categórica*, também chamada de *classificação* ou em *numérica*, também chamada de regressão. Cf. Silva, Peres e Boscarioli (2017) e Kesavaraj e Sukumaran (2013), também exposto em Ferrari e Silva (2017) e Goldschmidt e Passos (2005).

Formalmente, a tarefa de classificação pode ser descrita como a busca por uma função de mapeamento para um conjunto X de vetores de entrada (ou, exemplares — os dados) $\vec{x_i} \in E^d$ para um conjunto finito de rótulos C de cardinalidade c. A função F é, então, definida como $F : E^d \times W \rightarrow C$, em que d é a dimensão do espaço E, ou seja, a quantidade de coordenadas do vetor $\vec{x_i}$, e W é um espaço de parâmetros ajustáveis por meio do algoritmo de indução supervisionada (SILVA; PERES; BOSCARIOLI, 2017).

Pode ser dividida em, ao menos, duas categorias: *classificação binária* e *classificação multiclasse*. Na binária, a cardinalidade c é 2. Para o caso em que $c > 2$, o problema é considerado de múltiplas classes (SILVA; PERES; BOSCARIOLI, 2017) (KESAVARAJ; SUKUMARAN, 2013).

Os textos de Kesavaraj e Sukumaran (2013), Shazmeen, Mustafa e Baig (2013), além dos de Wolpert (1996), Kumar (2012) e Al-Radaideh e Nagi (2012) trazem reflexões, técnicas, comparações e explicações detalhadas de muitos algoritmos de classificação, entre eles, árvores de decisão, k-vizinhos mais próximos, Naive Bayes e Redes Bayesianas, Redes Neurais Artificiais, Máquinas de Vetores de Suporte (SVM) entre outros.

Sobre *teoria da aprendizagem* e *algoritmos de classificação* há uma discussão em Russell e Norvig (2013) sobre qual seria, em relação às hipóteses de modelos de aprendizagem, aquela (ou aquelas) que melhor se ajuste aos dados futuros. Os autores citam a ***suposição de estacionaridade***, ou seja, que há uma distribuição de probabilidade sobre os dados que permanece estacionária ao longo do tempo. Supõe-se, portanto que cada exemplo de ponto de dados (antes de conhecê-lo) é uma variável aleatória E_j cujo valor observado $e_j = (x_j, y_j)$ é amostrado da distribuição e é independente dos exemplos anteriores.

Assim:

$$P(E_j|E_{j-1}, E_{j-2}, ...) = P(E_j), \tag{2.2}$$

e cada exemplo tem uma distribuição de probabilidade anterior idêntica:

$$P(E_j) = P(E_{j-1}) = P(E_{j-2}) = \dots \tag{2.3}$$

Estes exemplos são chamados de *independentes e identicamente distribuídos* ou **i.i.d**. Esta suposição é, segundo os autores, necessária para *tentar a previsão sobre o futuro dos dados*. Há claro, ainda em Russell e Norvig (2013), um alerta sobre o fato de ser possível a aprendizagem ocorrer caso haja pequenas alterações (lentas) na distribuição.

Outro fato importante para a definição e avaliação da escolha da melhor hipótese (modelo) de um algoritmo de classificação é definir o "melhor ajuste". Em seu conhecido livro de Inteligência Artificial, os autores Russell e Norvig (2013) definem a **taxa de erro** de uma hipótese como uma métrica importante para definir o "melhor ajuste" de um modelo/hipótese.

2.5.0.1 Taxa de erro

A taxa de erro é, assim, a proporção de erros que o algoritmo classificador comete — a proporção de vezes que $h(x) \neq y$ para o exemplo (x, y) — sendo $h(x)$ a função que mapeia uma *hipótese/modelo* h com a *previsão/valor* conhecido y. Nem sempre, como alerta, Russell e Norvig (2013), uma hipótese/modelo h que tenha uma taxa de erro baixa no conjunto de treinamento generaliza bem e se comporta eficientemente para dados não conhecidos. A forma de testar o algoritmo é importante. Para isso há, na literatura, algumas técnicas que são utilizadas como estratégia de treinamento, validação e teste.

2.5.0.2 Estratégias de validação

Como citado por Russell e Norvig (2013) e Luger (2015) entre diversos autores e, cf. Silva, Peres e Boscarioli (2017, p. 125),

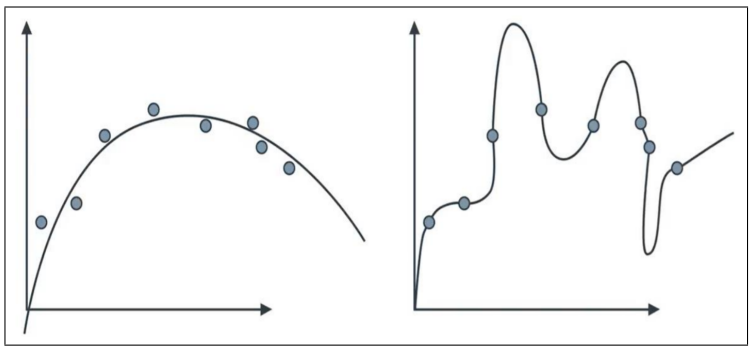
> independentemente da medida de avaliação a ser usada para atestar a qualidade de um modelo, não é adequado avaliá-lo [apenas] por seu desempenho em relação aos exemplares apresentados no processo de treinamento (indução). É sempre necessário saber como o modelo se comporta quando aplicado a exemplares que ainda não conhece, ou seja, não usados no processo de sintonização de seus parâmetros.

Essa observação, claro, é porque modelos preditivos, dependendo de como são criados, podem levar ao ***overfitting*** (sobreajuste). O fenômeno do sobreajuste ocorre, cf. Silva, Peres e Boscarioli (2017) "quando o modelo preditivo é gerado de forma a representar os exemplares usados para sua geração com uma fidelidade mais alta que o necessário".

Assim, modelos sobreajustados (ou superajustados) não são capazes de realizar predições adequadas para dados novos. Busca-se, na construção de modelos preditivos, uma maior *generalização* e no caso do sobreajuste o fenômeno contrário ocorre.

A maior causa de sobreajuste é quando os dados de treinamento não representam fielmente os dados novos (ou de produção) ou por serem diferentes (caso de dados antigos, por exemplo) ou por não serem significativos (poucos dados)[3]. A Figura 3 mostra um modelo preditivo em que o lado direito representa um modelo sobreajustado e o lado esquerdo um modelo regularizado para o mesmo *dataset*.

Figura 3 – Modelo sobreajustado e regularizado para o mesmo *dataset*.



Fonte: baseado no encontrado em Kasturi (2019)

Nas palavras de Amaral (2018, p. 95), "o processo de construção de um modelo de aprendizado de máquina busca, obviamente, maximizar a precisão e minimizar a taxa de erros". Para tanto, a literatura cita diversas técnicas e estratégias para a avaliação de modelos preditivos. Autores diversos, como Silva, Peres e Boscarioli (2017), Amaral (2016), além de Grus e Nascimento (2016) e Ferrari e Silva (2017) citam, geralmente, como estratégia de treinamento, validação e teste as seguintes técnicas:

[3] é possível também o sobreajuste devido a *ruídos* nos dados, pelo uso de um modelo de forma inapropriada ou de uma *classe rara* Amaral (2016, p.35)

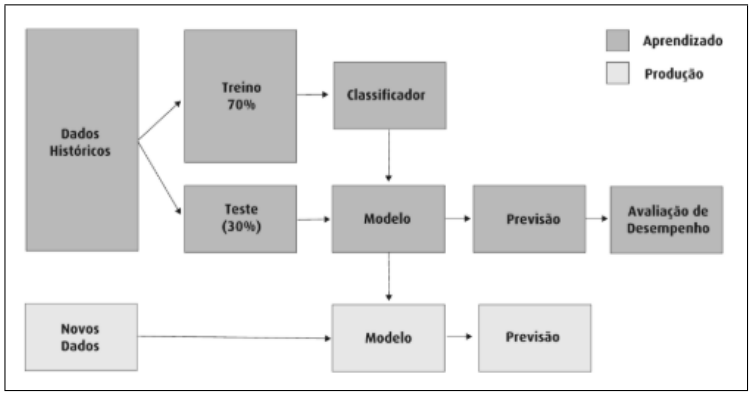
- Resubstituição;
- Holdout;
- Validação cruzada;
- Bootstrap;

Na **resubstituição**, segundo Silva, Peres e Boscarioli (2017), as medidas de avaliação dos classificadores são aplicadas no próprio conjunto de dados usados para indução do modelo. Essa técnica, embora tenha alguns vantagens discutidas em Ferrari e Silva (2017) e Silva, Peres e Boscarioli (2017), pode levar ao, já citado, sobreajuste (*overfitting*) e é discutido em Grus e Nascimento (2016), também em Amaral (2016) e Russell e Norvig (2013). Como já afirmado anteriormente, o sobreajuste é quando se produz um modelo de bom desempenho com os dados de treinamento, mas que não lida bem com novos dados.

Na técnica de **Holdout**, pressupõem-se uma divisão, ou criação de dois subconjuntos de dados distintos, a partir do conjunto de dados disponível pra uso na indução do modelo/hipótese. Um desses subconjuntos será usado para treinamento (indução) do modelo de previsão e o segundo, para teste após o término do treinamento e, consequentemente, na aplicação das medidas de avaliação do modelo/hipótese (SILVA; PERES; BOSCARIOLI, 2017)

A Figura 4 mostra o funcionamento da técnica de holdout de forma mais detalhada.

Figura 4 – Funcionamento da técnica holdout.



Fonte:(AMARAL, 2016)

Na estratégia de **validação cruzada**, todos os dados farão parte, em algum momento, do conjunto de dados usado no teste do modelo/hipótese. A ideia é que cada exemplo sirva duplamente — como dados de treinamento e dados de teste. Primeiro divide-se o conjunto em k subconjuntos iguais. Em seguida realiza-se k rodadas de aprendizagem; em cada iteração $\frac{1}{k}$ dos dados é retido como conjunto de teste e os exemplos restantes são usados como treinamento.

Valores populares de k são 5 e 10 — o suficiente para uma estimativa estatisticamente provável que seja precisa a um custo 5-10 vezes maior no tempo de computação. Há também o extremo do $k = n$, também conhecido como **validação cruzada com omissão de um**. O método de validação cruzada permite que o modelo/hipótese seja avaliado uma série de vezes, cada série sendo conhecida como partição (ou *fold*). Ao final, a avaliação pode ser realizada aplicando medidas estatísticas como média, desvio-padrão e intervalo de confiança ao conjunto de k avaliações obtidas ou somando-se os desempenhos obtidos pelos k modelos gerados e dividindo essa soma pelo número de exemplares original (RUSSELL; NORVIG, 2013)(SILVA; PERES; BOSCARIOLI, 2017)(FERRARI; SILVA, 2017) (AMARAL, 2016).

A Figura 5 mostra um exemplo didático de como funciona a validação cruzada. Os dados que não fazem parte do conjunto de teste em cada rodada são utilizados no treinamento.

Figura 5 – Funcionamento da técnica *cross validation*

Fonte: (GUFOSOWA, 2019)

Já a técnica de ***Bootstrap*** funciona de forma parecida à estratégia *holdout*. Ela também usa dois conjuntos, um de treinamento e outro para teste, porém durante o processo de formação dos subconjuntos, exemplares que já foram sorteados podem novamente serem contemplados, com probabilidade igual. É uma estratégia que permite, portanto, a reposição.

Neste trabalho, todos os algoritmos de classificação usados foram testados usando as técnicas de resubstituição, *holdout* (com taxas de 70-30, 75-25 e 60-40), além de *cross-validation* com 3, 5 e 10 folds.

Como explicado em Wolpert e Macready (1995) e Wolpert (1996) não existe um algoritmo de aprendizado superior a todos os demais quando considerados todos os problemas de classificação possíveis (teorema **NFL**, ou *No Free Lunch*), portanto, variações foram executadas nos experimentos em todas técnicas avaliadas, alterando-se os padrões para chegar a métricas e medidas de avaliação mais eficientes.

2.5.0.3 Medidas de avaliação

Para Castro e Braga (2011),

> Tradicionalmente, a métrica usada na avaliação e seleção de modelos de classificação é a acurácia (ou taxa de erro) estimada em relação a um dado conjunto de teste. Essa metodologia é justificada pela formulação padrão do problema do aprendizado supervisionado que visa a minimização da probabilidade do erro global.

Há, porém, conforme Silva, Peres e Boscarioli (2017), Amaral (2016), Kesavaraj e Sukumaran (2013) e Kumar (2012) diversas medidas usadas na avaliação de classificadores.

A que será usada neste trabalho (com o objetivo de minimizar a probabilidade do erro global) é a, já citada, acurácia ou taxa de classificações corretas.

A métrica acurácia é dada, portanto, por:

$$\text{Acurácia} = |y - f(\aleph) = 0|, \tag{2.4}$$

em que $|\cdot|$ representa a contagem de vezes em que $\cdot$ é verdadeiro, f é o modelo preditivo, $\aleph$ é o subconjunto de dados sob o qual o modelo está sendo avaliado, $f(\cdot)$ é a classificação fornecida pelo modelo preditivo para cada um dos exemplares (dos dados), e y é a classe esperada como resposta cf. Silva, Peres e Boscarioli (2017, p. 129).

A acurácia de um classificador também pode ser descrita em termos do **erro de generalização** ξ_g, e uma função de perda binária e, portanto, ser interpretada como a probabilidade de ocorrer uma classificação correta. Dessa forma:

$$\text{Acurácia}_g = 1 - \xi_g \tag{2.5}$$

Ou seja, a acurácia é, basicamente o número de acertos (positivos) divido pelo número total de exemplos. Será a métrica mais usada para avaliar os classificadores neste trabalho.

Há, entretanto, em modelos preditivos que trabalham com dados numéricos (como as redes neurais e o SVM), pode-se utilizar uma função de perda contínua, capaz de medir o erro entre a resposta obtida pelo modelo preditivo e a resposta aguardada (SILVA; PERES; BOSCARIOLI, 2017) (ACADEMY, 2020).

Para Reed e MarksII (1999)

> A função de custo reduz todos os aspectos bons e ruins de um sistema complexo a um único número, um valor escalar, o que permite ranquear e comparar as soluções candidatas.

Algumas funções de perda (*loss functions*), comuns, neste contexto são, de acordo com Silva, Peres e Boscarioli (2017):

$$Erro\ absoluto = \sum_{<\vec{x}_i, y_i> \in \aleph} |y_i - f(\vec{x}_i)| \tag{2.6}$$

$$Erro\ quadrático = \sum_{<\vec{x}_i, y_i> \in \aleph} (y_i - f(\vec{x}_i))^2 \tag{2.7}$$

$$Erro\ absoluto\ médio = \sum_{<\vec{x}_i, y_i> \in \aleph} |y_i - f(\vec{x}_i)|/m \tag{2.8}$$

$$Erro\ médio\ quadrático = \sum_{<\vec{x}_i, y_i> \in \aleph} (y_i - f(\vec{x}_i))^2/m \tag{2.9}$$

Onde m é a quantidade de instâncias existentes em $\aleph$.

Silva, Peres e Boscarioli (2017), Amaral (2018) além de Deng et al. (2016) e Ruuska et al. (2018) afirmam que é importante analisar o tipo de erro que o modelo está cometendo e, principalmente, no caso de classificadores binários, é possível observar que, mesmo com uma acurácia alta o classificador pode não estar respondendo de forma adequada.

Para realizar esse tipo de análise, os autores citados no princípio deste parágrafo, citam que uma ferramenta adequada é a ***matriz de confusão***.

Geralmente, uma matriz de confusão tem dimensões $C \times C$, em que C é o número de classes presentes no problema de classificação que está sendo avaliado. As linhas dessa matriz são indexadas seguindo as "classes esperadas" (y), e as colunas seguindo as "classes preditas" ($f(x) ou \hat{y}$).

Cada célula é um contador que é incrementado a depender do resultado da comparação de $f(x)$ e y. As respostas corretas do modelo geram os valores que entram na diagonal principal da matriz (SILVA; PERES; BOSCARIOLI, 2017, p. 130).

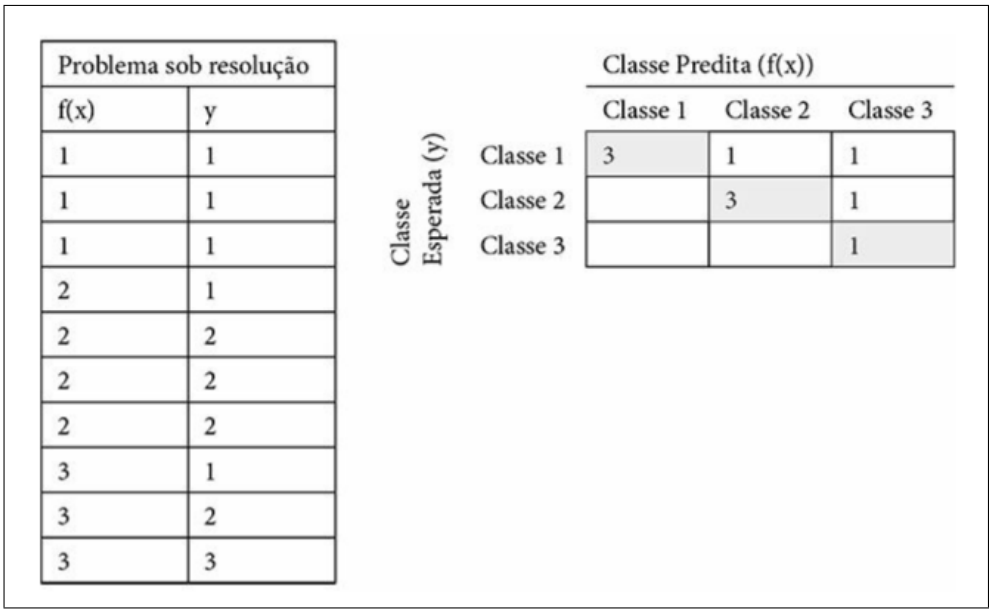
A Figura 6 demonstra o procedimento de preenchimento de uma matriz de confusão conforme a resolução de um problema hipotético de classificação.

Matrizes de confusão são, cf. Ruuska et al. (2018) e Silva, Peres e Boscarioli (2017), particularmente úteis para avaliação de classificadores binários. Neste caso, os valores são atribuídos conforme ilustrado na figura (neste caso, as duas classes estão definidas como "classe positiva" e "classe negativa").

Cada célula, neste caso, possui o seguinte significado:

- Verdadeiro Positivo (VP): classificação correta na classe positiva. A instância pertence a classe positiva e o modelo classificou na classe positiva.

Figura 6 – Preenchimento de uma matriz de confusão

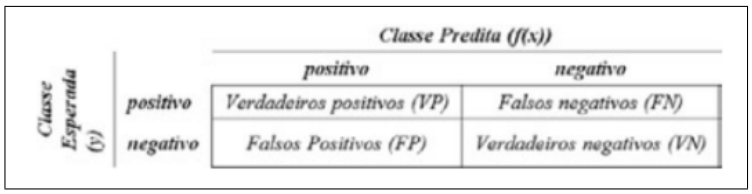


Problema sob resolução	
f(x)	y
1	1
1	1
1	1
2	1
2	2
2	2
2	2
3	1
3	2
3	3

Classe Esperada (y)	Classe Predita (f(x))		
	Classe 1	Classe 2	Classe 3
Classe 1	3	1	1
Classe 2		3	1
Classe 3			1

Fonte: Silva, Peres e Boscarioli (2017, p. 130)

Figura 7 – Matriz de confusão para o caso de um classificador binário



Classe Esperada (y)	*Classe Predita (f(x))*	
	positivo	*negativo*
positivo	*Verdadeiros positivos (VP)*	*Falsos negativos (FN)*
negativo	*Falsos Positivos (FP)*	*Verdadeiros negativos (VN)*

Fonte: Silva, Peres e Boscarioli (2017, p. 131)

- Falso Positivo (FP): classificação incorreta na classe negativa. A instância pertence à classe negativa, mas o classificador a classificou como pertencente à classe positiva.
- Verdadeiro Negativos (VN): classificação correta na classe negativa. A instância pertence a classe negativa e o modelo classificou na classe negativa.
- Falso Negativo (FN): classificação incorreta na classe negativa. classificação incorreta na classe negativa. A instância pertence à classe positiva, mas o classificador a classificou como pertencente à classe negativa.

Na próxima seção serão descritos alguns conceitos envolvendo as RNA's, Redes Neurais Artificiais, um dos (dois) classificadores que serão mais explorados neste trabalho, por motivos que serão explorados no Capítulo 3.

2.6 Redes Neurais Artificiais - RNA

As redes neurais instituem um campo da ciência da computação, parte da área da inteligência artificial, que busca efetivar modelos matemáticos que se assemelhem às redes

neurais biológicas. Elas apresentam capacidade de adaptar seus parâmetros como resultado da interação com o meio externo (FERNEDA, 2006)(RUSSELL; NORVIG, 2013).

2.6.1 Definição

Para Santos (2013, p. 41)

> Uma rede neural artificial consiste de um sistema composto por neurônios dispostos em camadas, interligados através de pesos sinápticos construindo um sistema que simula o cérebro humano, inclusive seu comportamento. E uma técnica computacional que apresenta um modelo inspirado na estrutura neural de organismos inteligentes e que adquire conhecimento através da experiência.

De acordo com Lima, Pinheiro e Santos (2016, p. 47), "redes neurais podem ser caracterizadas como modelos computacionais com capacidades de adaptar, aprender, generalizar, agrupar ou organizar dados".

Inicialmente, portanto, se desenvolveram como uma estratégia de simular os processos mentais humanos, como reconhecimento de imagens e sons, e após, como instrumento tecnológico e eficiente para muitas tarefas (JIN; CHEU; SRINIVASAN, 2002).

Para Obaidat e Macchairolo (1994), as redes neurais artificiais podem ser usadas efetivamente para prover soluções para um amplo espectro de aplicações, incluindo mapeamento de padrões e classificação, análise e codificação de imagens, processamento de sinais, otimização, manipulação de grafos, reconhecimento de caracteres, reconhecimento automático de alvo, fusão de dados, processamento de conhecimento, controle de qualidade, mercado de ações, processamento de hipotecas, triagem de créditos para empréstimos entre muitos outros problemas.

2.6.2 Modelo de neurônio artificial

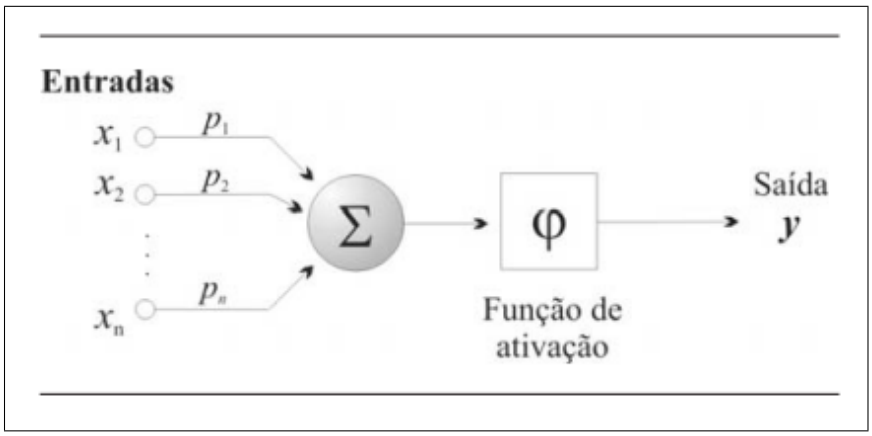
Desde a década de 1940 com o trabalho de Mcculloch e Pitts (1943) que se busca um modelo computacional que simule o cérebro humano e suas conexões. O interesse pela pesquisa nesta área cresceu e se desenvolveu durante os anos 50 e 60. É dessa época que Rosenblatt (1958) sugeriu um método de aprendizagem para as redes neurais artificiais chamado *perceptron*.

Até o final da década de 1960 muitos trabalhos foram feitos usando o percepton como modelo, mas ao final desta década, Minsky e Papert (1969) apresentaram significativas limitações do perceptron.

A pesquisa diminui consideravelmente nos anos seguintes (o chamado inverno da IA), porém durante os anos 80, a excitação ressurge mediante os avanços metodológicos importantes e, também, ao aumento dos recursos computacionais disponíveis. O modelo

de neurônio artificial da Figura 8 é uma simplificação do apresentado por Haykin (2001, p. 36).

Figura 8 – Modelo matemático de um neurônio



Fonte: Haykin (2001, p. 36)

O modelo da Figura 8 é composto por três elementos:

- um conjunto de n conexões de entrada $(x_1, x_2, \ldots, x_n)$, caracterizadas por pesos $(p_1, p_2, \ldots, p_n)$;
- um somador $(\sum)$ para acumular os sinais de entrada;
- uma função de ativação (φ) que, no caso específico do neurônio apresentado por McCullock-Pitts em Mcculloch e Pitts (1943) é uma função de limiar (FERNEDA, 2006) (LIMA; PINHEIRO; SANTOS, 2016).

O comportamento das conexões entre os neurônios é simulado através de seus pesos $(p_1, p_2, ..., p_n)$. Os valores podem ser positivos ou negativos (dependendo se a conexão é *inibitiva* ou *excitativa*.

O efeito de um sinal proveniente de um neurônio é determinado pela multiplicação do valor do sinal recebido pelo peso da conexão correspondente $(x_i \times p_i)$.

Então é efetuada a soma dos valores $x_i \times p_i$ de todas as conexões e o valor resultante é enviado para a função de ativação que define a saída (y) do neurônio. cf. Russell e Norvig (2013) e Mcculloch e Pitts (1943), além de Minsky e Papert (1969), Ferneda (2006) e Haykin (2001)

Matematicamente, a saída y_k do neurônio mostrado na Figura 8 é dada pela expressão abaixo:

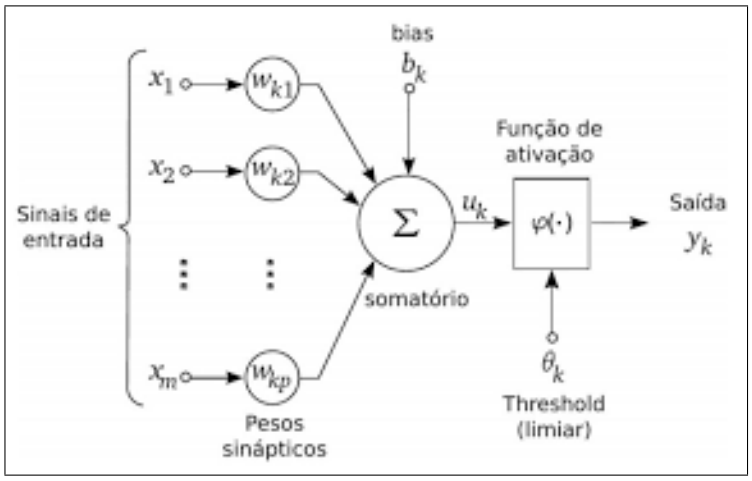
$$Y_k = \varphi(u_k) \qquad \text{onde} \qquad u_k = \sum_{i=1}^{n} P_{ki} X_i \tag{2.10}$$

A função de ativação proposta inicialmente por McCullock e Pitts em seu trabalho seminal, Mcculloch e Pitts (1943) é uma função de limiar:

$$y_k = \varphi(u_k) = \begin{cases} 1, & \text{se } u_k > 0 \\ 0, & \text{se } u_k < 0 \end{cases} \tag{2.11}$$

Uma alteração importante neste modelo foi a introdução de um parâmetro polarizador (*bias* ou *offset*) b_k, conforme a Figura 9 cujo objetivo é deslocar o valor da informação referente à entrada líquida u_k, de forma a verter a função de ativação no eixo correspondente ao valor de u_k. Assim a saída $y_k = \varphi(u_k)$.

Figura 9 – Adição de um *offset* (*bias*) no modelo do neurônio



Fonte: Lima, Pinheiro e Santos (2016, p. 58)

Dessa forma, o polarizador (bias) pode ser tratado como mais um peso da rede. Basta considerar um nova entrada do tipo $x_0 = 1$, com um peso associado $w_0 = b_k$, assim a representação fica sendo, considerando os pesos como w, de acordo com a Figura 9:

$$u_k = w_0 + w_1 x_1 + w_2 x_2; \qquad y_k = \varphi(u_k) \tag{2.12}$$

Assim, esse neurônio pode ser empregado, segundo, Lima, Pinheiro e Santos (2016, p. 58), para "separar classes distintas de padrões de entradas para aplicações de classificações de padrões". E prossegue: "se a entrada líquida for maior que o limiar, o padrão dessa entrada pertence à classe 1, caso contrário, pertence à classe 0". Analisando matematicamente o modelo *perceptron*, se percebe que ele pode ser considerado um típico caso de discriminador linear.

A fronteira de decisão para um *perceptron* como o da Figura 9, dada duas entradas e a função de ativação mostrada na Equação 2.11 será, então, uma reta cuja equação é

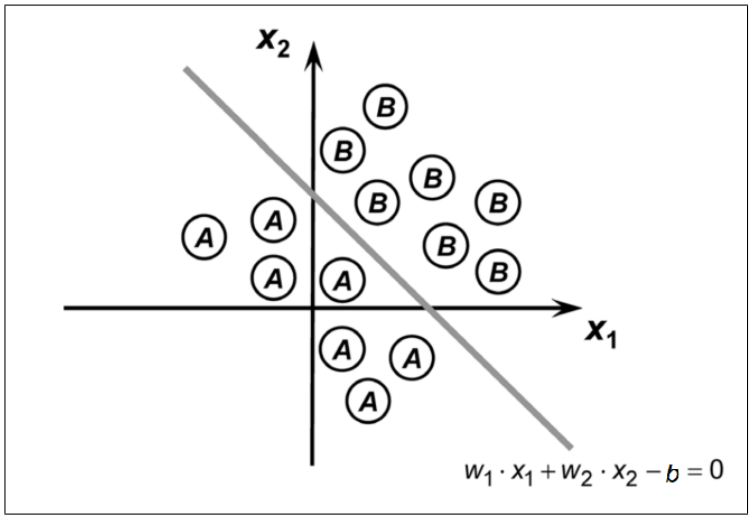
definida por:

$$w_1 x_1 + w_2 x_2 - b = 0 \tag{2.13}$$

Portanto, cf. Silva (2016), "pode-se concluir que o Perceptron se comporta como um classificador de padrões cuja função é dividir classes que sejam linearmente separáveis".

A Figura 10 mostra uma reta posicionada na fronteira de separação entre as classes. Para este tipo de problema o perceptron é um classificador adequado.

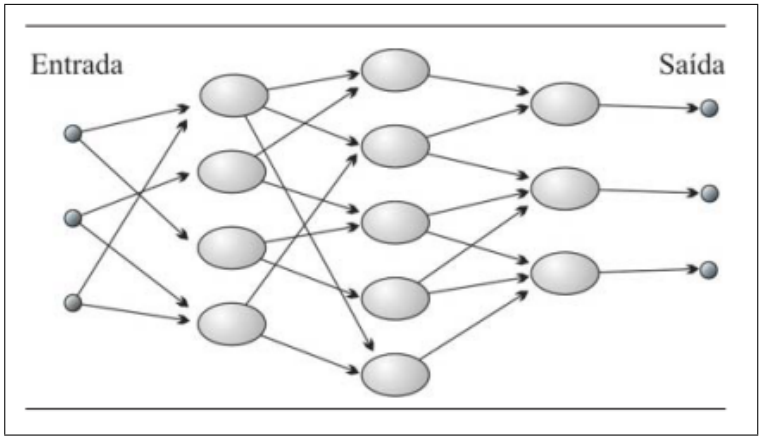
Figura 10 – Fronteira de separação (perceptron com duas entradas)



Fonte: Silva (2016, p. 62)

As redes neurais artificiais (**RNA**) se formam quando diversos neurônios se combinam. De forma resumida, "uma rede neural artificial (RNA) pode ser vista como um grafo onde os nós são os neurônios e as ligações fazem a função das sinapses". Isto está demonstrado na Figura 11.

Figura 11 – Representação simplificada de uma RNA



Fonte: Ferneda (2006, p.26)

As redes neurais artificiais se diferem pelas suas arquiteturas e pela forma como os pesos associados às conexões são ajustados durante o processo de aprendizado. A *arquitetura* de uma rede neural restringe o tipo de problema no qual a rede poderá ser utilizada, e é definida pelo número de camadas (camada única ou múltiplas camadas), pelo número de nós em cada camada, pelo tipo de conexão entre os nós (*feedforward* ou *feedback*) e por sua topologia (HAYKIN, 2001, p. 46-49).

O desenvolvimento de uma rede neural artificial consiste em determinar sua arquitetura, ou seja, os números de camadas e de neurônios em cada camada, bem como o ajuste dos pesos na fase conhecida como treinamento (HAGAN; DEMUTH; BEALE, 1996) (HAYKIN, 2001).

Uma das características mais importantes de uma rede neural artificial é a habilidade de aprender através de exemplos e fazer inferências sobre o que aprendeu, melhorando, assim, o seu desempenho. As RNA's utilizam um algoritmo de aprendizagem que serve, basicamente, para ajustar os pesos de suas conexões (HAYKIN, 2001) (FERNEDA, 2006) (LIMA; PINHEIRO; SANTOS, 2016) (RUSSELL; NORVIG, 2013).

Aqui também há, cf. explicitado na seção 2.4, duas formas básicas de aprendizado, o supervisionado e o não-supervisionado.

2.6.3 Redes do tipo Perceptron de múltiplas camadas

As arquiteturas do tipo perceptron de múltiplas camadas (MLP) são os modelos de redes neurais mais utilizados e conhecidos. Elas, basicamente, consistem de uma camada de entrada e uma ou mais camadas intermediárias (ou ocultas) além, claro da camada de saída (uma ou mais unidades sensoriais - neurônios) (HAYKIN, 2001).

Os sinais de entrada são propagados camada a camada pela rede em uma direção, ou seja, da entrada para a saída (*feedfoward*). Esta arquitetura retrata uma generalização do perceptron. Portanto, segundo Silva (2016, p. 26),

> As redes Perceptron de múltiplas camadas (PMC) são caracterizadas pela presença de pelo menos uma camada intermediária (escondida) de neurônios, situada entre a camada de entrada e a respectiva camada neural de saída.

Desta forma, elas possibilitam elevadas possibilidades de aplicações em muitas áreas do conhecimento, entre as principais: aproximação universal de funções, reconhecimento de padrões, identificação e controle de processos, previsões de séries temporais, otimização de sistemas entre muitos outros (HAYKIN, 2001).

A Figura 12 ilustra uma rede do tipo perceptron de multicamadas. O treinamento deste tipo de rede é do tipo supervisionado e, geralmente, se utiliza um algoritmo muito

popular chamado *retropropagação* do erro (*error backpropagation*). Este algoritmo é baseado numa regra de aprendizagem que "corrige" o erro durante o treinamento (HAYKIN, 2001).

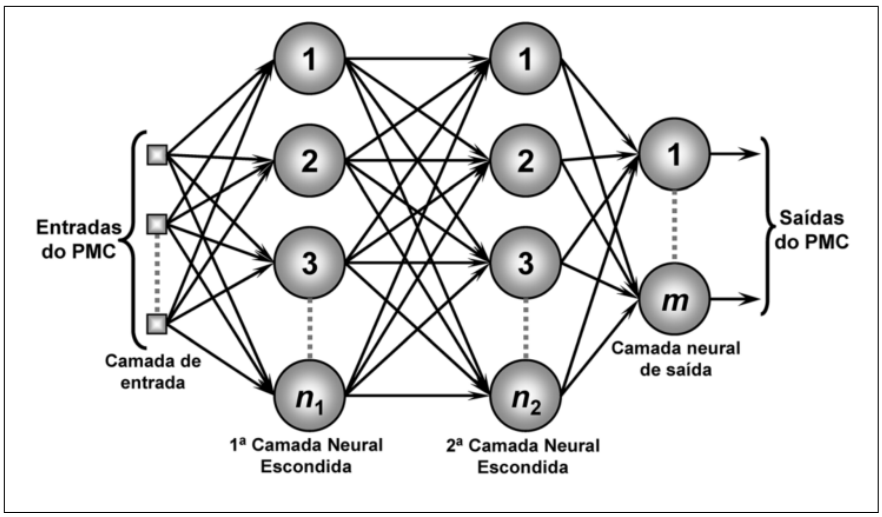
Substancialmente, o método de *retropropagação* é constituído de duas etapas: uma fase de propagação do sinal no sentido tradicional (*feedforward*) e uma de retropropagação do erro (*backpropagation*) de todas as camadas e seus respectivos pesos. Na fase de ida, os vetores de dados e pesos são aplicados às unidades de entrada, e seu efeito se propaga pela rede, camada por camada (HAGAN; DEMUTH; BEALE, 1996) (HAYKIN, 2001).

Após isso, um conjunto de saídas é produzido como resposta da rede. Na retropropagação os pesos são ajustados de acordo com uma regra de correção de erro (normalmente uma função matemática como a Equação 2.6, a Equação 2.8, a Equação 2.7 ou Equação 2.9, sendo esta última uma das mais utilizadas em muitas arquiteturas, na prática) (HAYKIN, 2001) (HAGAN; DEMUTH; BEALE, 1996) (YEUNG; BOTVINICK; COHEN, 2004).

A resposta da rede em um instante é subtraída da saída desejada (*target*) para produzir um valor de erro. Este valor de erro é propagado da saída para a entrada, camada a camada, de onde vem o nome "*retropropagação* do erro". Os pesos são, então, redefinidos de forma que a distância entre a resposta da rede e a resposta desejada seja reduzida (o erro seja minimizado). O processo é repetido diversas vezes até que uma tolerância global de erro seja assumida. Cada iteração é denominada época (*epoch*) (HAYKIN, 2001) (HAGAN; DEMUTH; BEALE, 1996) (MINSKY; PAPERT, 1969).

Uma arquitetura de um MLP (*Multi-Layer Perceptron* - Perceptron Multicamadas), possui, portanto três propriedades distintas: a função de ativação, o número de camadas ocultas e forma das conexões (totalmente conectada ou não).

Figura 12 – Rede Perceptron de multicamadas



Fonte: Silva (2016, p. 92)

2.6.4 Funções de ativação

Como visto na subseção 2.6.2, o modelo perceptron, lida bem com problemas linearmente separáveis, mas tem problemas ao lidar com problemas não-lineares (HAYKIN, 2001).

Para dar capacidade representativa às redes neurais artificiais, são essenciais as diferentes funções de ativação, pois só assim elas conseguirão lidar um componente de não-linearidade, como o são a maioria dos problemas práticos (HAGAN; DEMUTH; BEALE, 1996).

Ao se introduzir ativações não-lineares, a superfície de custo da rede neural deixa de ser convexa fazendo com que a otimização se torne mais difícil (MINSKY; PAPERT, 1969) (HAYKIN, 2001).

As principais funções de ativação, utilizadas na literatura ena prática, são, portanto:

2.6.4.1 Função de ativação limiar

Foi proposta na primeira definição de rede com neurônios artificiais (ROSENBLATT, 1958). O modelo de neurônio usado por RosenBlatt foi o mesmo sugerido por Mcculloch e Pitts (1943).

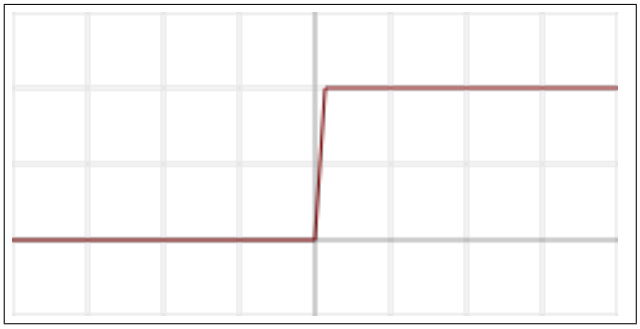
Neste modelo as saídas são binárias, ou seja, assumem, normalmente o valor 0 ou 1. A saída é 1 se o valor da entrada líquida for superior a um determinado valor chamado *thereshold*. Na maioria das vezes, esse valor é zero.

Matematicamente:

$$y_k = y = f_i(a_i(t)) = \begin{cases} 1, & \text{se } a_i(t) >= 0 \\ 0, & \text{se } a_i(t) < 0 \end{cases} \tag{2.14}$$

A Figura 13 mostra o 'funcionamento' da função de ativação limiar. Eventualmente podem ser utilizados os valores -1 e 1. Uma rede de camada simples que utiliza este tipo de ativação é o perceptron simples.

Figura 13 – Função de ativação Limiar

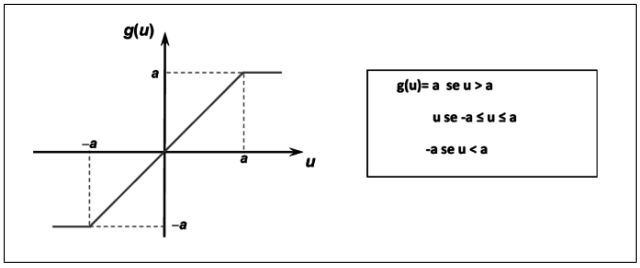


Fonte: Adaptado de Haykin (2001)

2.6.4.2 Função de ativação linear

A saída do neurônio, neste caso, é representada por uma função linear da forma descrita na Figura 14. Redes que usam este tipo de função de ativação apresentam apenas uma camada de entrada e uma de saída. Possuem, portanto, uma série de representações quanto ao que são capazes de representar (LIMA; PINHEIRO; SANTOS, 2016).

Figura 14 – Função de ativação Linear



Fonte: Adaptado de Haykin (2001)

2.6.4.3 Funções de ativação semilineares

As funções de ativação mais usadas deste tipo são a função logística e a tangente hiperbólica. Elas são populares por conta de suas derivadas (que são necessárias nas etapas de treinamento, particularmente no *Gradiente Descent* — descida do gradiente) poderem ser expressas a partir das próprias funções.

Estas funções, tanto a logística (ou *sigmoide*) e a tangente hiperbólica, respectivamente, podem ser expressas pelas equações 2.15 abaixo, onde a representa a entrada líquida da unidade (HAYKIN, 2001) (LIMA; PINHEIRO; SANTOS, 2016).

$$f(a) = \frac{1}{1 + e^{(-2\beta a)}} \qquad g(a) = tanh(\beta a) \tag{2.15}$$

E as respectivas derivadas das funções 2.15 acima são dadas por:

$$f'(a) = 2\beta f(1 - f) \qquad g'(a) = \beta(1 - g^2) \tag{2.16}$$

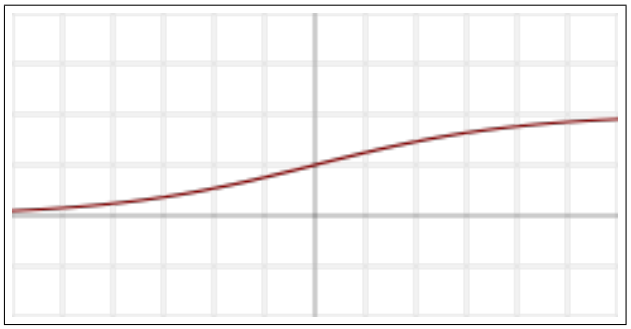
A Figura 15 mostra o comportamento da função sigmoide e a Figura 16 demonstra o da função tangente hiperbólica.

2.6.4.4 Função de ativação ReLu

A função ReLU é a unidade linear retificada. É definida como, Academy (2020):

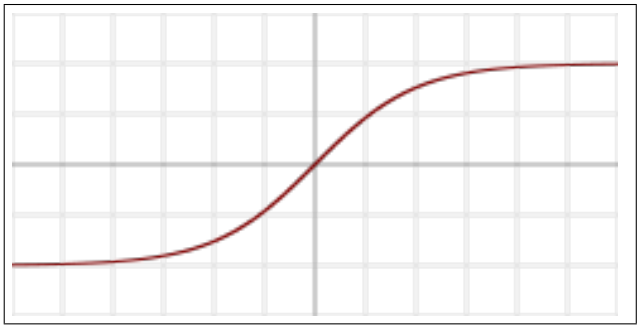
$$ReLU(x) = max(0, x) \tag{2.17}$$

Figura 15 – Função de ativação Logística (sigmoide)



Fonte: Adaptado de Haykin (2001)

Figura 16 – Função de ativação Tangente Hiperbólica



Fonte: Adaptado de Haykin (2001)

Sua derivada é dada por:

$$ReLU'(x) = \begin{cases} 1, & \text{se } x > 0 \\ 0, & \text{caso contrário} \end{cases} \tag{2.18}$$

Conforme, Academy (2020),

> ReLU é a função de ativação mais amplamente utilizada ao projetar redes neurais atualmente. Primeiramente, a função ReLU é não linear, o que significa que podemos facilmente copiar os erros para trás e ter várias camadas de neurônios ativados pela função ReLU.

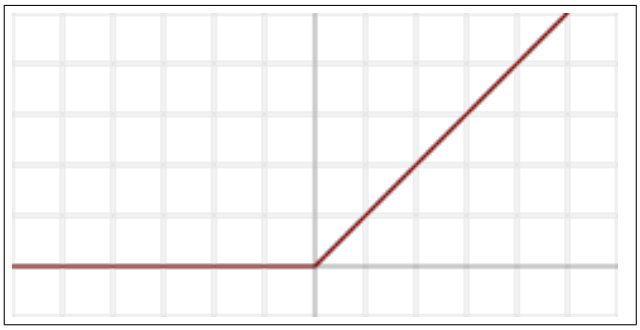
Ainda, de acordo com Academy (2020):

> A principal vantagem de usar a função ReLU sobre outras funções de ativação é que ela não ativa todos os neurônios ao mesmo tempo. [...] Se [...] a entrada for negativa, ela será convertida em zero e o neurônio não será ativado. Isso significa que, ao mesmo tempo, apenas alguns neurônios são ativados, tornando a rede esparsa e eficiente e fácil para a computação.

A Figura 17 demonstra o comportamento da função de ativação ReLu em função da entrada

Redes com a função ReLU são mais fáceis de otimizar porque a função é parecida com a função identidade. A única diferença é que a ReLU produz zero em metade de seu domínio (HAGAN; DEMUTH; BEALE, 1996).

Figura 17 – Função de ativação ReLu - unidade linear retificada



Fonte: Adaptado de Haykin (2001)

2.7 Otimização

Redes Neurais tem uma etapa iterativa em que os pesos são ajustados e se pode realizar otimizações para melhorar o erro e incrementar o modelo (HAYKIN, 2001).

Normalmente, o fluxo de treinamento se resume nos seguintes passos iterativos:

1. Operar a entrada na rede
2. Cálculo da função de perda (*loss function*) ou outra função de cálculo de erro
3. Cálculo do gradiente
4. Atualização dos pesos
5. Volta para o passo 1

Este processo é repetido até que um hiperparâmetro de tolerância seja alcançado para o cálculo dos erros mediante a *loss function* ou outra função de cálculo do erro.

O cálculo do gradiente descrito anteriormente diz respeito, cf. Academy (2020) a um dos mais usados algoritmos para otimizar a tarefa de aprendizagem dos pesos de uma rede neural.

De acordo com Academy (2020):

> A Descida do Gradiente é uma ferramenta padrão para otimizar funções complexas iterativamente dentro de um programa de computador. Seu objetivo é: dada alguma função arbitrária, encontrar um mínimo. Para alguns pequenos subconjuntos de funções – aqueles que são convexos – há apenas um único *minimum* que também acontece de ser global. Para as funções mais realistas, pode haver muitos mínimos, então a maioria dos mínimos são locais.[... É preciso] que a otimização encontre o "melhor" *minimum* e não fique preso em mínimos sub-otimistas (um problema comum durante o treinamento do algoritmo).

O algoritmo consiste basicamente de subtrair o valor do gradiente ∇f dos pesos w da rede, assim:

$$w_i = w_i - \alpha \times \nabla f_i \tag{2.19}$$

Sendo α o multiplicador que nos permite controlar o tamanho do passo de otimização. ∇f é a derivada da função de ativação no ponto específico. α é um hiperparâmetro conhecido como taxa de aprendizado e repreenta a *velocidade* em que a rede neural "aprende" os melhores pesos para o problema específico (HAYKIN, 2001).

Uma iteração consiste em um passo de otimização como o descrito em 2.7 e corresponde a uma ligação sináptica de *forward* na rede e uma de *backpropagation*. Isto constitui uma **época**. Normalmente, muitas épocas são necessárias, visto que o aprendizado em uma rede neural é, geralmente, lento para que a otimização evite os mínimos locais.

Na próxima seção, conceitos sobre SVM (Support Vector Machines) serão descritos e explorados. Este será o segundo algoritmo utilizado neste trabalho para sustentar a hipótese, juntamente com as RNA's.

2.8 SVM - Support Vector Machines

Segundo Cortes e Vapnik (1995), o algoritmo SVM (*Support Vector Machines*) é um dos mais efetivos para a tarefa de classificação.

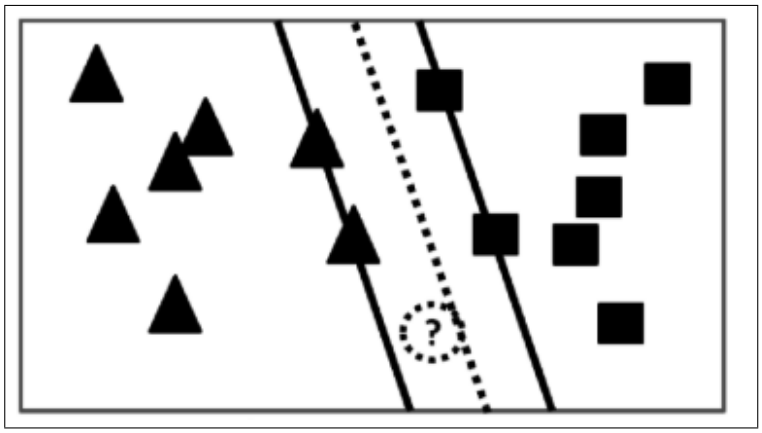
Cf. Goldschmidt e Passos (2005),

> No algoritmo SVM, o conjunto de dados de entrada é utilizado para construir uma *função de decisão* $f(x)$, tal que:
>
> $$\begin{aligned} &Se \quad f(x_i) \geq 0, \quad \text{então} \quad y_i = 1 \\ &Se \quad f(x_i) < 0, \quad \text{então} \quad y_i = -1 \end{aligned} \tag{2.20}$$
>
> O algoritmo SVM constrói os denominados classificadores lineares, que separam o conjunto de dados por meio de um *hiperplano* que é a generalização do conceito de *plano* para dimensões maiores que três.

Assim, SVM, cf. Amaral (2016, p. 45) "são um algoritmo de classificação que maximizam as margens entre instâncias mais próximas, dessa forma, é criado um vetor otimizado que é então utilizado para classificar novas instâncias".

Conforme se vê na Figura 18, os dois vetores *não pontilhados* são as margens otimizadas. As instâncias por onde as margens otimizadas passam são os vetores de suporte. O vetor pontilhado é a referência para classificar novas instâncias. Assim, a nova instância, na Figura 18 é classificada como triângulo.

Figura 18 – Vetores de Suporte



Fonte: Amaral (2016, p. 45)

Seguindo o estudo de Mukkamala, Janoski e Sung (2002) há duas razões principais que levaram os autores do artigo citado de usarem SVMs para detecção de intrusão:o primeiro é a velocidade já que a performance é prioritariamente uma das características mais importantes para sistemas de detecção de intrusos.

A segunda razão é a escalabilidade, pois, cf. os autores, SVMs são relativamente indiferentes ao número de *data points* e a complexidade da classificação não depende da dimensionalidade do espaço de características. Dependendo da aplicação, ainda conforme os autores, uma vez que os dados estão classificados em duas classes, um algoritmo de otimização adequado pode ser usado, se necessário, para identificação de mais características.

Neste trabalho, também foi usado, com eficácia que suportasse a hipótese (cf. se vê no Capítulo 3) o algoritmo SVM.

2.9 Trabalhos Relacionados

Esta seção expõe alguns dos principais trabalhos encontrados na literatura que estão, de alguma forma, relacionados a esta proposta de dissertação. Foram estudadas e analisadas obras acerca das políticas de segurança da informação, abordando a detecção de conflitos em diversos contextos como identificação e resolução de conflitos aéreos, rodoviários e de *normas* além de trabalhos sobre mineração de dados e técnicas de aprendizagem de máquina tanto em conjunturas distintas como regressão e classificação quanto para localização e solução de conflitos, no caso específico das políticas computacionais, principalmente sendo usadas em registros de patentes americanas fechadas — o que, de certa forma, evidencia o quanto este tema está na vanguarda.

Todos os trabalhos descritos foram base para o estudo, amadurecimento bibliográfico e aprofundamento teórico sobre o problema e as soluções propostas neste trabalho, principalmente aqueles relacionados a intrusões, detecção de conflitos aéreos e as soluções

baseadas em aprendizado de máquina, com ênfase nas redes neurais e nas máquinas de vetores de suporte (SVM) pois não só serviram de inspiração como ofereceram estímulo para que as técnicas pudessem ser extrapoladas para o uso na detecção de conflitos em políticas, tema desta proposta de dissertação.

2.9.1 Detecção e resolução de diversos tipos de conflitos

2.9.1.1 (LUPU; SLOMAN, 1999)

Os autores deste trabalho discorrem sobre conflitos no gerenciamento de sistemas distribuídos com base em políticas de controle de acesso. O artigo analisa os conflitos de políticas, concentrando-se nos problemas de detecção e resolução dos mesmos. Discute-se, no artigo, os vários relacionamentos de precedência que podem ser estabelecidos entre as políticas e é apresentado uma ferramenta de análise de conflitos que faz parte de uma estrutura de gerenciamento baseada em funções, porém, sem usar mineração de dados ou aprendizagem de máquina.

2.9.1.2 (CHEN, 2011) e (CHRISTODOULOU; KONTOGEORGOU, 2008)

O artigo de Chen (2011) trabalha com a identificação e a resolução de conflitos de voo com base em *redes neurais*. No artigo, o autor considera os problemas de detecção e resolução de conflitos no gerenciamento de tráfego aéreo (ATM - *Air Traffic Management*) sob a perspectiva da geometria computacional e fornece algoritmos para resolver esses problemas além de propor um método que pode rotear várias aeronaves, sem conflitos, através do espaço aéreo, usando um esquema de roteamento priorizado no espaço-tempo através do uso de conjunto de soluções ótimas por meio de uma rede neural. Já o trabalho de Christodoulou e Kontogeorgou (2008) aborda a detecção de conflitos sobre a ótica da prevenção de colisões no voo livre de aeronaves comerciais usando redes neurais com exemplos preparados por meio de programação não linear.

2.9.1.3 (SARKIS, 2017), (SILVESTRE, 2017) e (SARKIS; SILVA; BRAGA, 2016)

Estes são os trabalhos-base no estudo da detecção de conflitos, da determinação do modelo das políticas utilizadas e analisadas além de serem a fonte de algumas das principais definições empregadas neste trabalho. Silvestre (2017) estuda a verificação de conflitos entre múltiplas normas em sistemas multiagentes (SMA). As normas, nesta tese, são semelhantes às políticas, inclusive em suas definições, mas restritas a um contexto de agentes autônomos. Para a resolução de conflitos, o autor usa uma estratégia de aplicação de filtros para suavizar o custo computacional e utiliza transformação deôntica para análise de diversas normas ao mesmo tempo.

2.9.2 Mineração de dados e aprendizagem de máquina

2.9.2.1 (GUERRERO-HIGUERAS; DeCastro-GARCIA; MATELLAN, 2018)

Guerrero-Higueras, DeCastro-Garcia e Matellan (2018) aborda um método para construir modelos de *aprendizagem de máquina* com o objetivo de detectar ataques cibernéticos em RTLSs (*Real Time Location Systems* - sistemas de localização em tempo real) em um ambiente de cibersegurança para sistemas robóticos usando técnicas de *Machine Learning*. O artigo mostra que os ciberataques nos sistemas de localização em tempo real para sistemas robóticos podem ser detectados por um sistema criado usando o aprendizado supervisionado. Além disso, mostra que alguns tipos de ciberataques em sistemas de localização em tempo real, especificamente negação de serviço e falsificação(*DoS* e *Spoofing*), podem ser detectados por um sistema construído usando técnicas de aprendizado de máquina. Oito classificadores e algoritmos preditores conhecidos foram avaliados neste artigo e a análise de validação cruzada mostrou que os classificadores MLP (*Multi Layer Perceptron*) funcionam melhor que os outros obtendo maior acurácia e menor erro, sendo também o modelo com menor *overfitting* e maior *sensibilidade*[4]. Este artigo inspirou a hipótese desta proposta de dissertação.

2.9.2.2 (BUI; STOLLER; LE, 2019)

Bui, Stoller e Le (2019) versa acerca de mineração de dados em políticas de controle de acesso, especificamente do modelo ReBAC (*Relationship-Based Access Control* - controle de acesso baseado em *relacionamento*), mas não foca na detecção de conflitos entre as políticas, e sim, na propagação de seus relacionamentos na criação de novas políticas. No artigo, os algoritmos de mineração de política do ReBAC propostos puderam reduzir significativamente o custo da migração dos sistemas de controle de acesso legados para o ReBAC, automatizando parcialmente o desenvolvimento de uma nova política do ReBAC.

2.9.2.3 (OBAIDAT; MACCHAIROLO, 1994) e (MUKKAMALA; JANOSKI; SUNG, 2002)

Obaidat e Macchairolo (1994) aborda um sistema de rede neural multicamadas para segurança de acesso a computadores com o objetivo de identificar usuários do mesmo. Os vetores de entrada foram compostos pelos intervalos de tempo entre pressionamentos de teclas sucessivos criados pelos usuários ao digitar uma sequência conhecida de caracteres. Usando aprendizado supervisionado, cada vetor de entrada foi classificado em uma das várias classes, identificando assim o usuário que digitou a sequência de caracteres. Neste artigo, uma precisão máxima de classificação de 97,5% foi alcançada usando um classificador de padrões baseado em rede neural multicamadas *feedforward* treinada usando *backpropagation*, o algoritmo de retropropagação. Essa abordagem visa melhorar a segu-

[4] Sensibilidade é a proporção de verdadeiros positivos: a capacidade do sistema em predizer corretamente a condição para casos que realmente a têm - É também conhecida como ***Recall***.

rança do acesso ao computador. Este artigo trouxe contribuições para esta proposta de dissertação na maneira de construir a arquitetura da rede neural e também no ajuste dos *hiperparâmetros*. Na mesma linha, o artigo de Mukkamala, Janoski e Sung (2002) estuda a detecção de invasões usando redes neurais e máquinas de vetores de suporte e a ideia central é descobrir padrões úteis ou características que descrevam o comportamento intrusivo de um usuário em um sistema e os autores usam este conjunto de características para construir classificadores que puderam reconhecer anomalias e intrusões conhecidas em tempo real. É usado um conjunto de dados de referência de uma competição de KDD (*Knowledge Discovery in Databases* — Descoberta de Conhecimento em Bases de Dados) projetada pela DARPA (Defense Advanced Research Projects Agency), e é demonstrado que classificadores eficientes e precisos podem ser construídos para detectar invasões. Ao final é comparado o desempenho de redes neurais e máquinas de vetores de suporte para a detecção de intrusões. Nesta proposta de dissertação, baseando-se neste artigo de Mukkamala, Janoski e Sung (2002) também serão avaliados os desempenhos de redes neurais e máquinas de vetores de suporte, mas para detecção de conflitos em políticas.

2.9.2.4 (JIN; CHEU; SRINIVASAN, 2002) e (DEBAR; BECKER; SIBONI, 1992)

O trabalho de Jin, Cheu e Srinivasan (2002) aborda o desenvolvimento e uso de uma rede neural probabilística construtiva (**CPNN** - *Constructive Probabilistic Neural Network*) na detecção de incidentes em rodovias, incluindo a construção e adaptação dos modelos. Esta CPNN foi estruturada com base no modelo Gaussiano de mistura e treinada por um algoritmo de ajuste dinâmico de decaimento (para correção dos erros). Os incidentes em rodovias, como colisões de veículos, podem ser extrapolados para um modelo de conflito entre agentes de um sistema. O modelo foi treinado e avaliado sobre um banco de dados de incidentes simulados em Singapura e adaptado para a rodovia I-880 na Califórnia sendo então investigada em ambientes on-line e off-line. Este artigo citado compara o desempenho do modelo CPNN e um modelo de rede neural probabilística básica (BPNN - Basic Probabilistic Neural Network).Já o artigo de Debar, Becker e Siboni (1992) estuda um componente de rede neural para um sistema de detecção de intrusão. O modelo *aprende* os hábitos que um usuário tem enquanto trabalha com o computador e emite avisos quando o comportamento atual não é consistente com os padrões *aprendidos* anteriormente. O modelo de de rede neural é usado para modelar o comportamento do usuário como uma característica componente para o sistema de detecção de intrusão.

3 Experimentos/Resultados

Neste capítulo serão descritas as 3 abordagens diferentes que foram utilizadas e em cada uma delas, os principais algoritmos de classificação que foram utilizados. A primeira parte dos experimentos focaram em determinar quais os classificadores apresentaram melhor acurácia para a resolução do problema descrito na subseção 1.1.1. Em seguida, os dois que apresentaram melhor acurácia foram utilizados nos experimentos subsequentes.

3.1 Forma geral dos experimentos

A forma geral de como a estrutura dos experimentos foram realizados para o problema proposto na subseção 1.1.1 é a seguinte:

1. Definição do problema;
2. Coleta de dados;
3. Pré-processamento dos dados;
4. Engenharia e seleção de atributos;
5. Modelagem: definição, configuração e arquitetura da rede (ou modelagem dos hiper-parâmetros do SVM);
6. Treinamento (Aprendizagem da rede neural e do SVM);
7. Testes e validação do modelo;
8. Avaliação e ajuste do modelo;
9. Apresentação dos resultados;

Este modelo de método experimental foi adaptado daqueles propostos em Lima, Pinheiro e Santos (2016), Silva (2016) e Haykin (2001). O item 1 já foi descrito na subseção 1.1.1. As próximas seções trazem os passos 2 a 9.

3.2 Base de dados, pré-processamento e recursos computacionais

3.2.1 Experimentos iniciais - arquivo com 68 políticas

Para os experimentos iniciais, um arquivo de políticas foi gerado randomicamente a partir do proposto em Sarkis (2017) e, de acordo com o exposto na subseção 2.2.3.

O arquivo gerado possui, inicialmente, cerca de 68 políticas nomeadas (constituindo a *fase de seleção*[1] da Mineração de Dados) e da fase coleta de dado do método proposto anteriormente.

Este arquivo foi usado nos testes preliminares da hipótese descrita na subseção 1.1.3 deste trabalho. Para este problema da detecção de conflitos diretos serão usadas técnicas de aprendizagem supervisionada.

Para tanto, ao arquivo com as políticas, no *pré-processamento* foi acrescentada uma coluna rotulando os conflitos da seguinte forma: **1**: *conflito direto* e **0**: *sem conflito*. Esta estratégia é a mesma utilizada no trabalho de Davy, Jennings e Strassner (2008) onde ele usa um modelo de matriz de controle de acesso usando operações lógicas `AND` e `OR` para identificação de conflitos.

A figura 19 demonstra o aspecto do arquivo das políticas geradas para os experimentos deste trabalho. Na imagem, pode-se notar a *classe* (coluna) criada para guiar o aprendizado supervisionado dos algoritmos utilizados no estudo.

Figura 19 – Aspecto do arquivo das políticas geradas para os experimentos

```
Politica,Acesso,Organizacao,Sujeito,Acao,Objeto,DataAtivacao,DataDesativacao,Conflito
Policy01, Permitted,UFAC,Secretario_de_Curso_Academico,Abertura, Documentos, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy02, Forbidden,UFAC, Sandra_Maria_Soares_da_Rocha,Abertura, Documentos, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy03, Permitted,UFAC, null, Solicitar, Produtos, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy04, Forbidden,UFAC, Secretario_de_Centros_Academicos, Solicitar, Produtos, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy05, Permitted,UFAC, null,Acessar,Almoxarifado, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy06, Forbidden,UFAC, Jaider_Moreira_de_Almeida,Acessar,Almoxarifado, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy07, Permitted,UFAC, Secretarios, Solicitar, Materiais, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy08, Forbidden,UFAC, Secretario_de_Centros_Academicos, Solicitar, Materiais, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy09, Permitted,UFAC, Grupo_IPTU, Gerar, Planilhas_de_Calculo, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy10, Permitted,UFAC, Grupo_IPTU, Calcular, IPTU, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
Policy11, Forbidden,UFAC, Grupo_IPTU, Calcular, IPTU, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
Policy12, Permitted,UFAC, Socorro_Pontes,Acessar, Portal_do_Aluno, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy13, Permitted,UFAC, Socorro_Pontes, Matricular,Aluno, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy14, Permitted,UFAC, Jose_Rodrigues_Bardalles, Solicitacao, Central_de_Copias, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy15, Permitted,UFAC, Jose_Rodrigues_Bardalles, Analise, Central_de_Copias_Analise, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy16, Permitted,CCET, Jose_Rodrigues_Bardalles, Solicitacao, Central_de_Copias, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy17, Permitted,CCSJA, Jose_Rodrigues_Bardalles, Analise, Central_de_Copias, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy18, Permitted,UFAC, Grupo_Almoxarifado, Requisitar, Material, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy19, Forbidden,UFAC, Grupo_Almoxarifado, Criar, Guia_de_Requisicao, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy20, Forbidden,UFAC, Grupo_Almoxarifado, Inserir, Guia_de_Requisicao, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy21, Permitted,UFAC, Usuario_Qualquer, Cadastrar, Convencao, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
Policy22, Forbidden,UFAC, Usuario_Qualquer, Nova, Convencao, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
Policy23, Forbidden,UFAC, Usuario_Qualquer, Alterar, Convencao, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
Policy24, Forbidden,UFAC, Usuario_Qualquer, Cadastrar, Convencao, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
Policy25, Permitted,UFAC, Usuario_Qualquer, Nova, Convencao, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
Policy26, Permitted,UFAC, Usuario_Qualquer, Alterar, Convencao, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
Policy27, Permitted,UFAC, Socorro_Pontes, Solicitar, MatriculaAluno, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy28, Permitted,UFAC, Socorro_Pontes, Efetivar, MatriculaAluno, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy29, Permitted,UFAC, Joao_Josino, LancarMedia, Notas, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
Policy30, Permitted,UFAC, Joao_Josino, VoltarLancamento, Notas, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy31, Forbidden,UFAC, Joao_Josino, LancarMedia, Notas, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
Policy32, Permitted,UFAC, Joao_Josino, VoltarLancamento, Notas, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy33, Permitted,UFAC, Secretario_de_Unidade_Administrativa, SolicitacaoAbertura, Documentos, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
Policy34, Forbidden,UFAC, Secretario_de_Unidade_Administrativa, Cancelamento, Documentos, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
```

Fonte: compilação do autor

Ainda na fase de *pré-processamento*, a coluna 9 (Conflito) foi transformada do tipo de dado *Numérico para Nominal*. Para isso foi usado o softwate WEKA (descrito em Witten et al. (2016)) aplicado o filtro *NumericToNominal* do software. Além disso, tanto o primeiro atributo quanto a data foram removidos, pois, dentro do escopo estudado neste trabalho, eles não influenciariam nos resultados finais.

3.2.2 Recursos computacionais

Dois **ambientes computacionais** foram utilizados para as tarefas de mineração: um **notebook** Intel Core i5 vPro-8350U (8ª Geração de 64 bits com 1.70GHz e 8 GB de

[1] cf. seção 2.3 deste trabalho.

RAM, com SSD de 256 GB rodando Windows 10 Pro.

O outro ambiente foi um **Desktop** Intel Core i7 vPro-6700 de 8ª geração de 64 bits com 3.40 Ghz e 20 GB de RAM, com HD de 1 TB rodando o Windows 10 Pro.

3.2.3 Resultados - Arquivo com 68 políticas

Logo após, mais de 30 experimentos foram realizados de forma preliminar no *dataset* envolvendo os diversos algoritmos e muitos parâmetros alterados (a maioria com pequena ou nenhuma variação) para se chegar às técnicas finais que foram utilizadas nos posteriores experimentos e que serão explicitadas a seguir.

Utilizando-se a ferramenta WEKA descrita em (WITTEN et al., 2016) para as últimas fases da Mineração de Dados, foram utilizados alguns algoritmos de classificação que segundo Wu et al. (2007) são alguns dos mais utilizados na Mineração de Dados. Para avaliar o desempenho definiu-se o método *cross-validation* com 10 folds. Em seguida suas acurácias foram comparadas.

A tabela 1 mostra o resultado destes experimentos:

Tabela 1 – Acurácia dos classificadores

Classificador/Algoritmo	Acurácia
Multi Layer Perceptron	0.9705
SVM kernel linear	0.9705
Random Forest	0.9542
J48	0.9411
K* (K-star)	0.9411
Trees LMT	0.9117
IBk (KNN, com k =1)	0.8970
JRip	0.8970
Nayve Bayes	0.8674
Random Tree	0.7794

Fonte: Elaborada pelo autor mediante experimentos

As figuras 20 e 21 mostram os resultados das classificações do arquivo de políticas usando, respectivamente, os classificadores/algoritmos: *SVM* e o *MultiLayer Perceptron* (que foram os principais citados nos trabalhos relacionados, cf. descrito na seção 2.9). Cf. mostrado na Tabela 1, os classificadores SVM e MultiLayerPerceptron ficaram empatados em relação à acurácia. Nos experimentos posteriores, *apenas os dois foram considerados*.

Assim, com uma acurácia de 97,05% na classificação dos conflitos diretos, tanto o algoritmo Multilayer Perceptron (que implementa uma rede neural sigmoide multicamadas) quanto o SVM tiveram a maior acurácia, com 95,7% de *TP rate*(taxa de *True Positives* ou

Figura 20 – Saída do software WEKA. Classificador: SVM

```
=== Classifier model (full training set) ===

LibSVM wrapper, original code by Yasser EL-Manzalawy (= WLSVM)

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          66               97.0588 %
Incorrectly Classified Instances         2                2.9412 %
Kappa statistic                          0.9344
Mean absolute error                      0.0294
Root mean squared error                  0.1715
Relative absolute error                  6.6784 %
Root relative squared error             36.5941 %
Total Number of Instances               68

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,957    0,000    1,000      0,957    0,978      0,936    0,978     0,986     0
                 1,000    0,043    0,917      1,000    0,957      0,936    0,978     0,917     1
Weighted Avg.    0,971    0,014    0,973      0,971    0,971      0,936    0,978     0,964

=== Confusion Matrix ===

  a  b   <-- classified as
 44  2 |  a = 0
  0 22 |  b = 1
```

Fonte: compilação do autor

Figura 21 – Saída do software WEKA. Classificador: *MultiLayer Perceptron*

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          66               97.0588 %
Incorrectly Classified Instances         2                2.9412 %
Kappa statistic                          0.9344
Mean absolute error                      0.0574
Root mean squared error                  0.1664
Relative absolute error                 13.029  %
Root relative squared error             35.5106 %
Total Number of Instances               68

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,957    0,000    1,000      0,957    0,978      0,936    0,996     0,998     0
                 1,000    0,043    0,917      1,000    0,957      0,936    0,996     0,992     1
Weighted Avg.    0,971    0,014    0,973      0,971    0,971      0,936    0,996     0,996

=== Confusion Matrix ===

  a  b   <-- classified as
 44  2 |  a = 0
  0 22 |  b = 1
```

Fonte: compilação do autor

verdadeiros positivos) para a classe 0 (não há conflito) e, somente, 4,3% de *FP rate*(taxa de Falsos Positivos) para a classe 1 (quando há conflito direto). Nos experimentos realizados (assim como se esperava inicialmente na hipótese deste trabalho — baseado em evidências da literatura), estes modelos algorítmicos foram os mais eficientes para a detecção de conflitos diretos.

A interface visual do WEKA é interessante para observar o comportamento inicial dos algoritmos, mas para as arquiteturas de redes neurais e seus diversos parâmetros, configurações e quantidade de camadas ocultas, entre outras configurações levaram a outros experimentos com outras abordagens. Para isso e com o objetivo de evitar o *overfitting*

(como explicado na subseção 2.5.0.3 novos procedimentos e ferramentas foram adotados).

3.3 Outros experimentos - arquivos com 139 e 281 políticas

Foi gerado, então, dois novos arquivos de políticas, também randomicamente, e, ainda, a partir do proposto em Sarkis (2017) e, cf. o exposto na subseção 2.2.3. Os arquivos gerados agora, possui, 139 e 281 políticas nomeadas.

Os mesmos filtros anteriores foram aplicados e os resultados para os algoritmos (ainda usando a ferramenta Weka) foram os seguintes:

Para o MultiLayer Perceptron, a Tabela 2 mostra como as acurácias ficaram com os arquivos com quantidades diferentes de políticas.

Tabela 2 – Acurácia do MLP

Qtd. de Políticas	Acurácia
68	97.05
139	98.73
281	99.28

Fonte: Elaborada pelo autor mediante experimentos

Para o classificador SVM, a Tabela 3 mostra como as acurácias ficaram com os arquivos com quantidades diferentes de políticas.

Tabela 3 – Acurácia do SVM

Qtd. de Políticas	Acurácia
68	97.05
139	96.40
281	99.28

Fonte: Elaborada pelo autor mediante experimentos

Mesmo com uma leve queda na acurácia do SVM no arquivo de 139 políticas, percebe-se que, quanto maior o número de políticas, maior é a acurácia do classificador e, portanto, melhor é a classificação.

3.3.1 Experimentos com Pandas, NumPy e sklearn

Outros experimentos foram realizados utilizando as bibliotecas Pandas, descrita em McKinney (2010), NumPy, pormenorizada em Oliphant (2006) e sklearn, detalhado em Pedregosa et al. (2011) além do Notebook Jupyter, caracterizado em Kluyver et al. (2016).

O notebook criado demonstra o treinamento de uma rede neural Rede Neural Multicamadas usando o classificador **`MLPClassifier`** da biblioteca **`sklearn`**.

No arquivo, é demonstrado os experimentos na construção de uma arquitetura de uma rede neural com a estratégia de testes *holdout* sendo a divisão da base (split) em atributos previsores e classe com cerca de 75% da base sendo usada para treinamento da rede e 25% para teste.

Há também um pré-processamento importante focado em um tratamento dos dados categóricos do *dataset* e sua conversão para dados numéricos e divisão da base original em dois conjuntos de dados (previsores e classe).

Todos os atributos categóricos do dataset de políticas (o maior, com 281 instâncias) foram transformados para numéricos sendo um dicionário de dados construído.

Os passos foram os seguintes:

Primeiramente a base foi importada da seguinte forma, como demonstrado na Figura 22 mostrando o aspecto inicial do dataset.

Figura 22 – Aspecto do dataset importado

```
In [4]: base.shape
Out[4]: (281, 6)

In [5]: base.head()
Out[5]:
```

	Acesso	Organizacao	Sujeito	Acao	Objeto	Conflito
0	Permitted	UFAC	Secretario_de_Curso_Academico	Abertura	Documentos	0
1	Forbidden	UFAC	Sandra_Maria_Soares_da_Rocha	Abertura	Documentos	0
2	Permitted	UFAC	Outro_Usuario	Solicitar	Produtos	0
3	Forbidden	UFAC	Secretario_de_Centros_Academicos	Solicitar	Produtos	0
4	Permitted	UFAC	Outro_Usuario	Acessar	Almoxarifado	0

Fonte: compilação do autor

A partir daí, foram selecionados da base todas as linhas dos atributos do *dataset* que estavam com os tipos *object*. Foram procurados valores nulos e não foram encontrados.

Foi utilizada uma técnica que transforma um atributo categórico com k valores em uma representação numérica com valores inteiros para cada k valor. Há vantagens e desvantagens nessa abordagem. Elas estão discutidas em detalhes em Sarkar, Bali e Sharma (2017).

A Figura 23 demonstra como este procedimento foi realizado para o atributo que representa o acesso (Permitido, Proibido e Obrigatório).

Em seguida a base foi dividida em atributos previsores e a classe. Os previsores são as colunas que representam a política em si e a classe é a representação binária do conflito.

Figura 23 – Engenharia de atributos - dados categóricos textuais

```
In [15]: #Usando dicionarios para trocar os valores das colunas
         acesso = {"Acesso": {"Permitted":1, "Forbidden": 2, "Obliged": 3}}
         obj_df.replace(acesso, inplace=True)
         obj_df.head()
```

Out[15]:

	Acesso	Organizacao	Sujeito	Acao	Objeto
0	1	UFAC	Secretario_de_Curso_Academico	Abertura	Documentos
1	2	UFAC	Sandra_Maria_Soares_da_Rocha	Abertura	Documentos
2	1	UFAC	Outro_Usuario	Solicitar	Produtos
3	2	UFAC	Secretario_de_Centros_Academicos	Solicitar	Produtos
4	1	UFAC	Outro_Usuario	Acessar	Almoxarifado

Fonte: compilação do autor

Figura 24 – Aspecto dos atributos previsores

```
In [39]: previsores
```

Out[39]:

	Acesso	Organizacao	Sujeito	Acao	Objeto
0	1	5	25	13	5
1	2	5	38	13	5
2	1	5	2	4	11
3	2	5	33	4	11
4	1	5	2	8	12
...	...	...	...	...	...
276	1	2	6	1	2
277	1	8	6	1	2
278	2	2	16	8	16
279	1	2	5	1	20
280	2	2	29	9	1

281 rows × 5 columns

Fonte: compilação do autor

O aspecto dos atributos previsores ficou como o mostrado na Figura 24. Já a classe ficou com o aspecto da Figura 25.

Em seguida os atributos foram transformados usando padronização para manter as variáveis na mesma ordem de grandeza. Na padronização, a média se iguala a 0 e o desvio-padrão se mantém em 1. A fórmula da padronização é a seguinte:

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

onde μ é a média aritmética e σ é o desvio-padrão dos dados. Como todos os dados estão agora em formato numérico, é um passo importante.

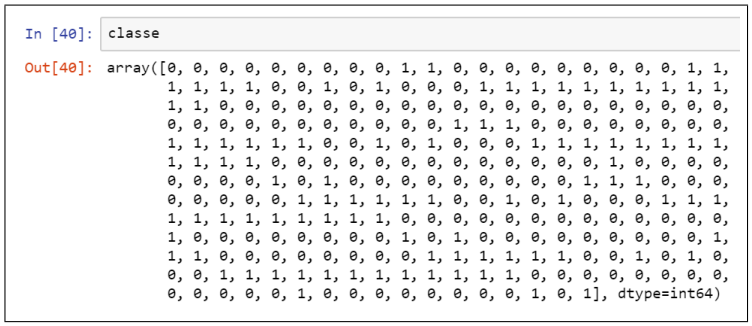
O código para a padronização foi o Código 3.1, abaixo:

Código 3.1 – Código da Padronização

```
10  from sklearn.preprocessing import StandardScaler
11  scaler = StandardScaler()
12  previsores_transformados = scaler.fit_transform(previsores)
```

Assim os dados foram padronizados e ficaram na mesma escala. Foi gerado um novo conjunto de dados de treinamento com 75% do *dataset* (sendo escolhidos randomicamente). Deixando, assim, 25% da base para validação.

Figura 25 – Aspecto do atributo classe



Fonte: compilação do autor

Logo em seguida, um modelo de Multi-Layer Perceptron foi criado usando o classificador **`MLPClassifier`** da biblioteca **`sklearn`** com os seguintes hiperparâmetros, mostrados no Código 3.2:

Código 3.2 – Hiperparâmetros do MLPClassifier

```
10  max_iter=10000,
11  tol = 0.0000010,
12  solver = 'adam',
13  hidden_layer_sizes=(100),
14  shuffle=False,
15  activation='relu'
```

Onde, respectivamente estão configuradas, o número máximo de épocas de treinamento (10000), a tolerância (0.0000010), a função de otimização de peso (‘adam’ refere-se a um otimizador estocástico baseado em gradiente descendente), a quantidade de neurônios na única camada oculta, se as amostras devem ser embaralhadas em cada iteração (marcado como falso) e a função de ativação da camada oculta (função de ativação ReLU).

A Figura 26 mostra o código no notebook e as iterações finais onde o classificador indica que a função de perda no treinamento não melhorou mais do que a tolerância, $tol = 0,000001$ por 10 épocas consecutivas e, portanto, ele encerrou as iterações.

Figura 26 – Código do **MLPClassifier** com as últimas iterações

```
import random
random.seed(42)
from sklearn.neural_network import MLPClassifier
classificador = MLPClassifier(verbose = True,
                              max_iter=10000,
                              tol = 0.0000010,
                              solver = 'adam',
                              hidden_layer_sizes=(100),
                              shuffle=False,
                              activation='relu')
classificador.fit(previsores_treinamento, classe_treinamento)
```

```
Iteration 4143, loss = 0.01138099
Iteration 4144, loss = 0.01160489
Iteration 4145, loss = 0.01174808
Iteration 4146, loss = 0.01174473
Iteration 4147, loss = 0.01157006
Iteration 4148, loss = 0.01140509
Iteration 4149, loss = 0.01138531
Iteration 4150, loss = 0.01144552
Iteration 4151, loss = 0.01150888
Iteration 4152, loss = 0.01146306
Iteration 4153, loss = 0.01137697
Training loss did not improve more than tol=0.000001 for 10 consecutive epochs. Stopping.
```

Fonte: compilação do autor

A Figura 27 mostra algumas métricas de validação após o modelo criado realizar as predições na base de teste (25% do dataset ou 71 políticas). Nela, pode-se perceber a acurácia do classificador em 95.77%, classificando, conforme a matriz de confusão mostrada na mesma figura, somente 3 previsões incorretas. Estes resultados apresentam uma sensibilidade (*recall*) de 97.7%. Levando-se em conta que o modelo se aprimora conforme a quantidade de instâncias aumenta, de acordo com o pressuposto no *teorema da aproximação universal* Hagan, Demuth e Beale (1996), pode se considerar este como um modelo satisfatório.

Figura 27 – Validações para o modelo **MLPClassifier**

```
previsoes = classificador.predict(previsores_teste)
```

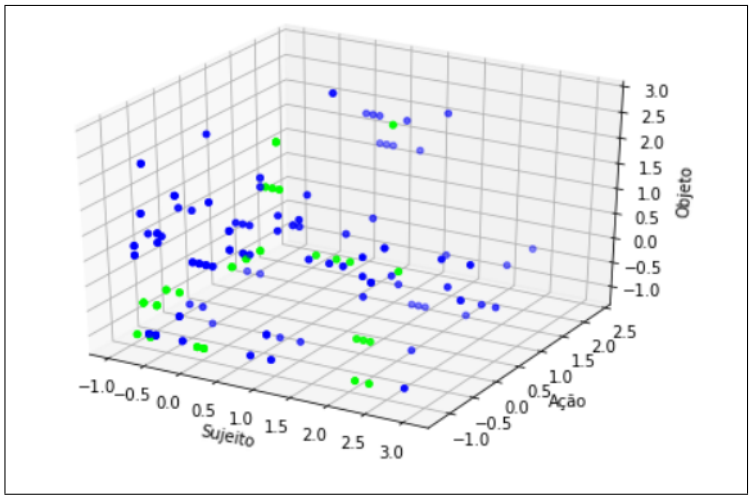
```
from sklearn.metrics import confusion_matrix, accuracy_score
precisao = accuracy_score(classe_teste, previsoes)
matriz = confusion_matrix(classe_teste, previsoes)
print(matriz)
print('Precisão: {}%'.format(precisao * 100))
```

```
[[43  2]
 [ 1 25]]
Precisão: 95.77464788732394%
```

Fonte: compilação do autor

A Figura 28 mostra a dimensionalidade dos dados após transformados pelo processo de padronização e dá uma visão geral de 3 atributos, ***sujeito, ação e objeto*** e o impacto na difusão dos conflitos no espaço após os atributos categóricos terem sido, todos, transformados para uma representação numérica.

Figura 28 – Dimensionalidade dos dados: atributos sujeito, ação e objeto



Fonte: compilação do autor

3.3.2 Experimentos com TensorFlow e Pytorch

De acordo com Géron (2019, p. 233), o *TensorFlow* é uma biblioteca de software para cálculo numérico de código aberto especialmente adequada e ajustada para o Aprendizado de Máquina em larga escala. Foi desenvolvido pela *Google Brain Team* para uso intensivo de redes neurais profundas. O código fonte foi disponibilizado em 2015 e está à disposição no link: `https://github.com/tensorflow/tensorflow`.

É possível usar computação paralela em várias CPU's ou GPU's além de suportar computação distribuída para que seja possível o treinamento e o uso de redes neurais em grandes conjuntos de treinamento dividindo os cálculos por centenas de servidores em um período de tempo razoável (GÉRON, 2019).

Entre suas características importantes destacam-se: rodar em diversos sistemas operacionais (e inclusive na nuvem); diversas APIs públicas para criação, treinamento e avaliação de arquiteturas de diferentes tipos de redes neurais; diversas outras API's de alto nível foram construídas com base no TensorFlow como o *Keras* e o *Pretty Tensor*; implementações em C++ altamente eficientes para operações de Aprendizado de Máquina; nós de otimização avançados para procura por parâmetros que minimizem uma função de custo; usa extensões CUDA, uma API destinada a computação paralela, GPGPU, e computação heterogênea, criada pela Nvidia que dá acesso ao conjunto de instruções virtuais da GPU e a elementos de computação paralela (GÉRON, 2019).

Os cálculos no *TensorFlow* são expressos como grafos de fluxo de dados , seu nome deriva das operações que as redes neurais realizam em arranjos de dados multidimensionais, chamados de "tensores"(generalização matemática de escalares, vetores e matrizes) (KADIMISETTY, 2018).

Foi realizado, assim, no escopo deste trabalho experimentos com a base de dados

de com 281 políticas já citada anteriormente. O notebook completo está disponível online no endereço: `https://bit.ly/33BmCzt`. Foi todo construído no ambiente Google Collab que é um serviço de nuvem gratuito hospedado pelo Google para incentivar a pesquisa de Aprendizado de Máquina e Inteligência Artificial, similar ao Jupyter Notebook, é uma lista de células que podem conter textos explicativos ou códigos executáveis e suas saídas (SANTOS, 2020).

Neste modelo os hiperparâmetros principais, cf. Thenmozhi, Kalaivani e Aravindan (2018) foram ajustados como: tamanho do batch em 20 (analisa, computa e reajusta os pesos de 20 instâncias de cada vez, por época), o número de GPU's trabalhando em conjunto para 4, a taxa de aprendizado em 0.00001, o decaimento dos pesos em 0.000005 e o número de épocas padrão em 30 (apenas para testes inciais).

Figura 29 – Separação dos dados de teste e treino

```
[78]  1 torch.manual_seed(1)
      2 indices = torch.randperm(len(dados_com_rotulos)).tolist()   # a
      3
      4 #separando os dados de teste e treino
      5 train_size = int(0.7*len(dados_com_rotulos))
      6 df_train   = dados_com_rotulos.iloc[indices[:train_size]]
      7 df_test    = dados_com_rotulos.iloc[indices[train_size:]]
      8
      9 print('Treino : ', len(df_train), ' - Teste: ', len(df_test))
     10 display(df_test.head())
     11
     12 df_train.to_csv('politicas_train.csv', index=False)
     13 df_test.to_csv('politicas_test.csv', index=False)
     14
     15 !ls

    Treino :  196  - Teste:  85
```

Fonte: compilação do autor

Para o processo de validação, usou-se o *holdout* com a separação em 70% das instâncias para o treinamento da rede neural e 25% para teste, predição e validação. Na figura 29 pode-se visualizar como o processo foi realizado,inclusive com a quantidade de instâncias em cada conjunto de dados (na linha 2 é feita a randomização do *dataset* original para evitar o overfitting e balancear a probabilidade da distribuição).

O pacote `torch.util.data` do PyTorch possui a classe abstrata `Dataset`. Ela permite que seja implementado o próprio dataset reescrevendo os métodos:

- `__init__(self)`: Define a lista de amostras do dataset
- `__getitem__(self, idx)`: Carrega uma amostra, aplica as devidas transformações e retorna uma tupla (dado, rótulo)
- `__len__(self)`: Retorna a quantidade de amostras do dataset

Dessa forma, a Figura 30 mostra a elaboração de uma classe chamada `Politicas` que implementa uma classe-filha que herda da superclasse, *Dataset* descrita acima.

Figura 30 – Implementação da classe Politicas

```
class Politicas(Dataset):
  def __init__(self, csv_path):
    self.dados = pd.read_csv(csv_path).to_numpy()

  def __getitem__(self, idx):

    sample = self.dados[idx][0:5]    # [2:14] são as colunas do dataset
    label  = self.dados[idx][-1:]     # [-1:]

    # converte pra tensor
    sample = torch.from_numpy(sample.astype(np.float32))
    label  = torch.from_numpy(label.astype(np.float32))

    return sample, label

  def __len__(self):
    return len(self.dados)
```

Fonte: compilação do autor

Em seguida, dois objetos DataLoader são criados, um para a base de treinamento e um para a base de teste. Em seguida, a Rede Neural Multicamadas é instanciada mediante a criação de uma classe chamada MLP que herda da classe `nn.Module` que representa um módulo genérico de uma rede neural.

A arquitetura da rede é configurada dentro da classe que a cria sendo: uma *camada linear* de entrada, duas *camadas lineares ocultas com 32 neurônios* em cada camada usando a função de ativação ReLU e *uma camada linear de saída com dois neurônios*, representando os dois rótulos do atributo que é a classe, neste caso específico, o atributo binário **conflito** que será predito.

É criada, no mesmo código da classe MLP, uma função que faz o avanço (fed-forward) das computações na rede e, ao final, é instanciada uma variável chamada **`net`** com as variáveis descritas:

- 5 atributos/neurônios na camada linear de entrada;
- 2 camadas ocultas com 32 neurônios cada; e
- 2 camadas de saída representando as variáveis preditas

Na mesma linha que cria o objeto `net` é feito o *cast* da rede na GPU para que ela possa, ao ser treinada, fazer uso dos poderes computacionais em paralelo da API, CUDA. A Figura 31 mostra o código descrito aqui.

Em seguida é definida uma *loss function*(função de perda ou de custo). É criado um critério que mede o erro médio quadrático (norma matricial ao quadrado) entre cada elemento na entrada x e o destino y, cf. Equação 2.9 da página página 41.

O otimizador utilizado é o Adam, um algoritmo para otimização estocástica descrito em Kingma e Ba (2017) passando para o algoritmo os valores da taxa de aprendizado e do decaimento de pesos mostrado anteriormente nesta seção.

Figura 31 – Implementação da classe que modela a arquitetura da rede

```
class MLP(nn.Module):

  def __init__(self, input_size, hidden_size, out_size):
    super(MLP, self).__init__()

    self.features  = nn.Sequential(
                       nn.Linear(input_size, hidden_size),
                       nn.ReLU(),
                       nn.Linear(hidden_size, hidden_size),
                       nn.ReLU()
                     )
    self.out     = nn.Linear(hidden_size, out_size)

  def forward(self, X):

    feature = self.features(X)
    output  = self.out(feature)

    return output

input_size  = len(train_set[0][0])    # quantidade de atributos *importantes*
hidden_size = 32
out_size    = 2                       # variaveis que serão preditas

net = MLP(input_size, hidden_size, out_size).to(args['device']) #cast na GPU
```

Fonte: compilação do autor

O fluxo de treinamento desta arquitetura de rede neural multicamadas proposta neste experimento segue o algoritmo iterativo:

- Iterar nas épocas
- Iterar nos batches (a quantidade de instâncias simultâneas)
- Cast dos dados no dispositivo de hardware (GPU)
- Forward na rede e cálculo da *loss function*
- Cálculo do gradiente e atualização dos pesos

Esse conjunto de passos é responsável pelo processo iterativo de otimização de uma rede. A validação, entretanto, é apenas a aplicação da rede em dados nunca antes vistos para estimar a qualidade do modelo no mundo real.

Três funções, portanto, são criadas, uma para modelar o treinamento da rede neural, uma para o teste e validação e outra que agrega as duas primeiras em uma só para executar o algoritmo descrito anteriormente. Então, para finalizar o experimento com o TensorFlow, o PyTorch, CUDA e o Google Collaboratory, *foi realizado um treinamento com 500 épocas da rede neural* explanada nesta seção e os resultados tanto do treino quando do teste e validação foram armazenados.

As figuras 32, 33 e 34 mostram as três funções citadas anteriormente.

Figura 32 – Implementação da função de treino da rede

```
def train(train_loader, net, epoch):
  net.train()

  epoch_loss = []
  for batch in train_loader:

    dado, rotulo = batch

    # Cast na GPU
    dado   = dado.to(args['device'])
    rotulo = rotulo.to(args['device'])

    # Forward
    pred = net(dado)
    loss = criterion(pred, rotulo)
    epoch_loss.append(loss.cpu().data)

    # Backward
    loss.backward()
    optimizer.step()

  epoch_loss = np.asarray(epoch_loss)

  print("Epoca %d, Loss: %.4f +\- %.4f" % (epoch, epoch_loss.mean(), epoch_loss.std()) )

  return epoch_loss.mean()
```

Fonte: compilação do autor

Figura 33 – Implementação da função de teste da rede

```
def test(test_loader, net, epoch):

  net.eval()
  with torch.no_grad():
    epoch_loss = []
    for batch in test_loader:

      dado, rotulo = batch

      # Cast na GPU
      dado   = dado.to(args['device'])
      rotulo = rotulo.to(args['device'])

      # Forward
      pred = net(dado)
      loss = criterion(pred, rotulo)
      epoch_loss.append(loss.cpu().data)

    epoch_loss = np.asarray(epoch_loss)

    print('********** Validate **********')
    print("Epoca %d, Loss: %.4f +\- %.4f" % (epoch, epoch_loss.mean(), epoch_loss.std()) )

    return epoch_loss.mean()
```

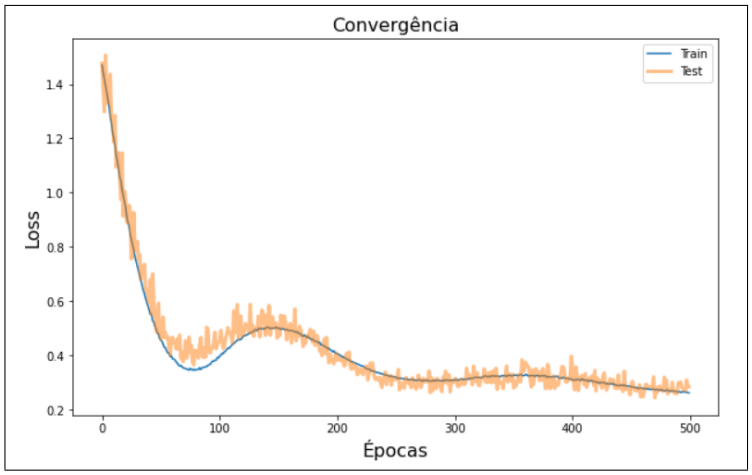
Fonte: compilação do autor

Figura 34 – Implementação da função que mescla o treino e o teste em uma só

```
def forward(loader, net, epoch, mode):
  if mode == "train":
    net.train()
  else:
    net.eval()

  epoch_loss = []
  for batch in loader:
    dado, rotulo = batch

    # Cast na GPU
    dado   = dado.to(args['device'])
    rotulo = rotulo.to(args['device'])

    # Forward
    pred = net(dado)
    loss = criterion(pred, rotulo)
    epoch_loss.append(loss.cpu().data)

    if mode == "train":
      # Backward
      loss.backward()
      optimizer.step()

  epoch_loss = np.asarray(epoch_loss)
  print("Epoca %d, Loss: %.4f +\- %.4f" % (epoch, epoch_loss.mean(), epoch_loss.std()) )
```

Fonte: compilação do autor

Figura 35 – Convergência das épocas entre o treino e o teste da MLP

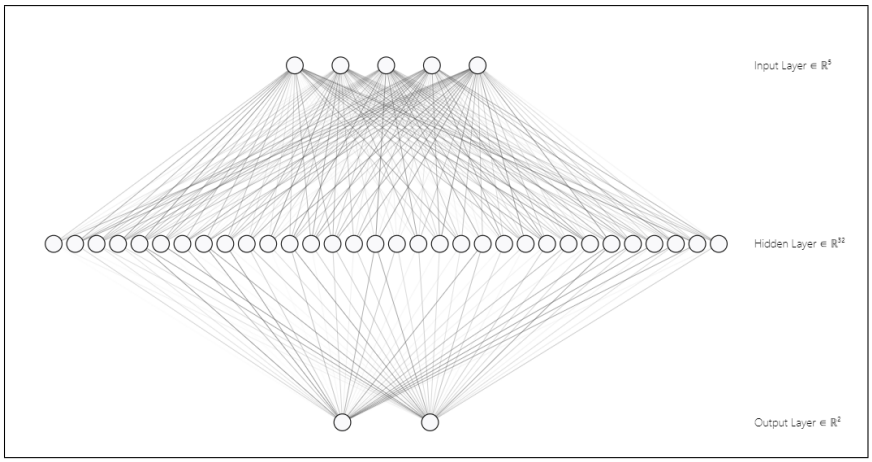


Fonte: compilação do autor

Como esclarecimento e interpretação visual foi construída, pois, a Figura 35 com os dados de armazenados de treino e teste (média da *loss function* ou função de custo de cada iteração dentro da época) que mostra um comparativo das épocas de teste e treino da rede neural e a convergência de ambas, retratando a acurácia e a validade do modelo deste experimento.

A arquitetura da rede neural construída e mostrada na Figura 31 pode ser visualizada na Figura 36 abaixo:

Figura 36 – Arquitetura da rede neural



Fonte: compilação do autor

3.3.3 Análise dos resultados

Os experimentos realizados na Capítulo 3 corroboram a hipótese de que a detecção de conflitos pode ser transformada em um problema da tarefa de classificação da mineração de dados e do aprendizado de máquina. As vantagens sobre as outras abordagens relatadas na literatura são:

- A detecção do conflito pode ser realizada em tempo de execução, pois os modelos já estão treinados;
- As políticas são verificadas "em lote" e não em pares já que a análise em pares é um problema computacionalmente custoso da classe NP-Completo como demonstrado por Shoham e Tennenholtz (1995);
- Ambas as técnicas analisadas neste trabalho mostraram-se eficazes com acurácias acima da proposta inicialmente (95%) cf. o mostrado na seção 3.3, seção 3.3, subseção 3.3.1 e subseção 3.3.2;
- os modelos de machine learning desenvolvidos nos experimentos podem ser aplicados em outros contextos, pois são suficientemente genéricos para tal;
- Ao analisar uma nova política, não é necessário "varrer" ou consultar todas as instâncias do *dataset* novamente já que os modelos já estão treinados;
- os modelos de machine learning tem a tendência a melhorarem a eficácia à medida que a quantidade de instâncias cresce

Todas os modelos e algoritmos demonstrados nos experimentos tiveram acurácia acima de 95% o que mostra que a detecção de conflitos em políticas (ou normas) pode ser colocada como uma classe de problemas a serem resolvidos de forma eficiente por técnicas de aprendizagem de máquina.

4 Cronograma e propostas para o texto final

Para a pesquisa que resultará na dissertação de mestrado os seguintes pontos serão levantados, estudados e melhor definidos em termos dos objetivos do trabalho:

1. Finalização da pesquisa teórica sobre o relacionamento de entidades;
2. Finalização da pesquisa teórica sobre funções de custo, SVM, descida do gradiente, topologias de redes neurais, suporte matemático aprofundado para o SVM;
3. estudo de outras redes, como Adaline, Madaline, redes de Kohonen, redes RBF e compará-las com as MLP e o SVM ;
4. Análise matemática profunda das outas função de ativação no classificador;
5. Análise teórica e implementação de outros otimizadores;
6. Realizar mais experimentos redes neurais e SVM;
7. Comparação final com outros classificadores (preferencialmente, geométricos, como o KNN e o SVM, avaliando suas acurácias e eficiência (para os conflitos indiretos).
8. Explorar a detecção dos conflitos indiretos usando Machine Learning e Aprendizagem de Máquina;

Para isso, propõem-se o seguinte cronograma descrito na tabela 4

Tabela 4 – Cronograma de finalização da dissertação

Atividades	**1ª qui ago**	**2ª qui ago**	**1ª qui set**	**2ª qui set**	**1ª qui out**	**2ª qui out**	**1ª qui nov**	**2ª qui nov**	**1ª qui dez**
Finalização da pesquisa teórica sobre o relacionamento de entidades	X	X							
Finalização da pesquisa teórica sobre funções de custo, SVM, descida do gradiente, topologias de redes neurais, suporte matemático aprofundado para o SVM;	X	X							
Estudo de outras redes, como Adaline, Madaline, redes de Kohonen, redes RBF e compará-las com as MLP e o SVM		X	X	X					
Análise matemática profunda das outras função de ativação no classificador			X	X					
Análise teórica e implementação de outros otimizadores			X	X	X				
Realizar mais experimentos redes neurais e SVM			X	X	X				
Comparação final com outros classificadores (preferencialmente, geométricos, como o KNN e o SVM,avaliando suas acurácias e eficiência (para os conflitos indiretos)				X	X	X			
Escrita, revisão e entrega de resultados iniciais					X	X	X	X	
Texto final com resultados							X	X	X

5 Conclusões

- Esta pesquisa mostrou a possibilidade de *converter a detecção de conflitos a um problema de classificação* especificamente, para os conflitos ***diretos***;
- O classificador mais acurado, nos experimentos, foi, como se imaginava pela hipótese, o *MultiLayer Perceptron* que é um classificador que usa *backpropagation* para aprender usando perceptron de várias camadas para classificar instâncias desconhecidas (WITTEN et al., 2016);
- Será realizada uma comparação entre o MLP e o SVM (e outros classificadores geométricos). Suas acurácias serão devidamente comparadas juntamente com a eficiência das soluções propostas.
- Todas os modelos e algoritmos demonstrados nos experimentos tiveram acurácia acima de 95% o que mostra, portanto, que a detecção de conflitos em políticas pode ser colocada como uma classe de problemas a serem resolvidos de forma eficiente por técnicas de aprendizagem de máquina.

Referências

ACADEMY, D. S. *Deep Learning Book, 2019*. Acesso em: 06 de março de 2020., 2020. (disponível em: http://deeplearningbook.com.br/). Capítulos em: http://deeplearningbook.com.br/capitulos/. Disponível em: <http://deeplearningbook.com.br/>. Citado 5 vezes nas páginas 32, 40, 50, 51 e 52.

AL-RADAIDEH, Q. A.; NAGI, E. A. Using Data Mining Techniques to Build a Classification Model for Predicting Employees Performance. In: . [S.l.: s.n.], 2012. Citado na página 36.

ALECRIM, E. *O que é Big Data?* INFOWESTER. Disponível em: https://www.infowester.com/big-data.php. Acesso em 12/12/2019, 2019. Disponível em: <https://www.infowester.com/big-data.php>. Citado 2 vezes nas páginas 16 e 23.

ALPAYDIN, E. *Introduction to Machine Learning, Third Edition*. 3ª. ed. The MIT Press, 2014. (Adaptive Computation and Machine Learning). ISBN 978-0-262-02818-9. Disponível em: <https://mitpress.ublish.com/ereader/26/?preview#page/Cover>. Citado na página 33.

AMARAL, F. *Aprenda Mineração de Dados: Teoria e prática*. ALTA BOOKS, 2016. (Autoria Nacional). ISBN 9788576089889. Disponível em: <https://books.google.com.br/books?id=qZIgDQAAQBAJ>. Citado 9 vezes nas páginas 30, 34, 35, 37, 38, 39, 40, 53 e 54.

AMARAL, F. *Introdução à Ciência de Dados: Mineração de dados e big data*. [S.l.]: Alta Books, 2018. Citado 3 vezes nas páginas 35, 37 e 41.

AUTREL, F.; COMA, C.; AL, e. MotOrBAC 2: a security policy tool. 2008. Citado na página 29.

BAEZA-YATES, R.; RIBEIRO-NETO, B. *Recuperação de Informação: Conceitos e Tecnologia das Máquinas de Busca*. Edição: 2. [S.l.]: Bookman, 2013. Citado 2 vezes nas páginas 33 e 34.

BUI, T.; STOLLER, S. D.; LE, H. Efficient and Extensible Policy Mining for Relationship-Based Access Control. In: *Proceedings of the 24th ACM Symposium on Access Control Models and Technologies*. New York, NY, USA: ACM, 2019. (SACMAT '19), p. 161–172. ISBN 978-1-4503-6753-0. Event-place: Toronto ON, Canada. Disponível em: <http://doi.acm.org/10.1145/3322431.3325106>. Citado 5 vezes nas páginas 15, 16, 25, 26 e 56.

CASACA, J. A.; CORREIA, M. F. Porque é necessária a segurança da informação? da estratégia às políticas de segurança. v. 0, n. 3, p. 89–116, 2013. ISSN 1647-1342. Number: 3. Disponível em: <http://revistas.lis.ulusiada.pt/index.php/lpis/article/view/157>. Citado 2 vezes nas páginas 23 e 24.

CASTRO, C. L. d.; BRAGA, A. P. Supervised learning with imbalanced data sets: an overview. v. 22, n. 5, p. 441–466, 2011. ISSN 0103-1759. Publisher: Sociedade Brasileira de

Automática. Disponível em: <http://www.scielo.br/scielo.php?script=sci_abstract&pid=S0103-17592011000500002&lng=en&nrm=iso&tlng=pt>. Citado na página 40.

CHEN, M. Q. Flight Conflict Detection and Resolution Based on Neural Network. In: *2011 International Conference on Computational and Information Sciences*. [S.l.: s.n.], 2011. p. 860–862. ISSN: null. Citado 3 vezes nas páginas 15, 18 e 55.

CHRISTODOULOU, M. A.; KONTOGEORGOU, C. Collision avoidance in commercial aircraft Free Flight via neural networks and non-linear programming. *International Journal of Neural Systems*, v. 18, n. 5, p. 371–387, out. 2008. ISSN 0129-0657. Citado 3 vezes nas páginas 15, 18 e 55.

CORTES, C.; VAPNIK, V. Support-vector networks. v. 20, n. 3, p. 273–297, 1995. ISSN 1573-0565. Disponível em: <https://doi.org/10.1007/BF00994018>. Citado na página 53.

CUPPENS, F.; CUPPENS-BOULAHIA, N.; GHORBEL, M. B. High level conflict management strategies in advanced access control models. v. 186, p. 3–26, 2007. ISSN 15710661. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1571066107004562>. Citado 2 vezes nas páginas 27 e 28.

DAVY, S.; JENNINGS, B.; STRASSNER, J. Application domain independent policy conflict analysis using information models. In: *NOMS 2008 - 2008 IEEE Network Operations and Management Symposium*. [S.l.: s.n.], 2008. p. 17–24. ISSN: 2374-9709. Citado na página 59.

DEBAR, H.; BECKER, M.; SIBONI, D. A neural network component for an intrusion detection system. In: *Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy*. [S.l.: s.n.], 1992. p. 240–250. ISSN: null. Citado 2 vezes nas páginas 15 e 57.

DENG, X. et al. An improved method to construct basic probability assignment based on the confusion matrix for classification problem. v. 340-341, p. 250–261, 2016. ISSN 0020-0255. Disponível em: <http://www.sciencedirect.com/science/article/pii/S002002551600044X>. Citado na página 41.

DOHERTY, N. F.; FULFORD, H. Do information security policies reduce the incidence of security breaches: An exploratory analysis. v. 18, n. 4, p. 21–39, 2005. ISSN 1040-1628. Disponível em: <https://doi.org/10.4018/irmj.2005100102>. Citado na página 24.

DUNLOP, N.; INDULSKA, J.; RAYMOND, K. Dynamic conflict detection in policy-based management systems. In: *Proceedings. Sixth International Enterprise Distributed Object Computing*. [S.l.: s.n.], 2002. p. 15–26. ISSN: null. Citado 2 vezes nas páginas 26 e 29.

ELRAKAIBY, Y.; CUPPENS, F.; CUPPENS-BOULAHIA, N. Formal enforcement and management of obligation policies. v. 71, n. 1, p. 127–147, 2012. ISSN 0169-023X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0169023X11001248>. Citado na página 27.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. v. 17, n. 3, p. 37–37, 1996. ISSN 2371-9621. Disponível em: <https://www.aaai.org/ojs/index.php/aimagazine/article/view/1230>. Citado 5 vezes nas páginas 16, 23, 30, 31 e 35.

FERNEDA, E. Redes neurais e sua aplicação em sistemas de recuperação de informação. v. 35, n. 1, 2006. ISSN 1518-8353. Disponível em: <http://revista.ibict.br/ciinf/article/view/1149>. Citado 4 vezes nas páginas 43, 44, 46 e 47.

FERRAIOLO, D. F. et al. Proposed NIST standard for role-based access control. v. 4, n. 3, p. 224–274, 2001. ISSN 1094-9224. Disponível em: <https://doi.org/10.1145/501978.501980>. Citado na página 25.

FERRARI, D. G.; SILVA, L. N. d. C. *Introdução a mineração de dados*. Editora Saraiva, 2017. ISBN 9788547200992. Disponível em: <https://books.google.com.br/books?id=SSlrDwAAQBAJ>. Citado 7 vezes nas páginas 16, 30, 31, 35, 37, 38 e 39.

FONG, P. W. Relationship-based access control: protection model and policy language. In: *Proceedings of the first ACM conference on Data and application security and privacy*. Association for Computing Machinery, 2011. (CODASPY '11), p. 191–202. ISBN 978-1-4503-0466-5. Disponível em: <https://doi.org/10.1145/1943513.1943539>. Citado na página 25.

FONTES, E. *Políticas e Normas para a Segurança da Informação*. [S.l.]: Brasport, 2012. Google-Books-ID: X61rbEWwJ_UC. ISBN 978-85-7452-515-0. Citado 2 vezes nas páginas 23 e 24.

FUGINI, M.; BELLETTINI, C. *Information Security Policies and Actions in Modern Integrated Systems*. Idea Group Pub., 2004. ISBN 978-1-59140-186-5. Disponível em: <https://books.google.com.br/books?id=uTgd_lruELcC>. Citado na página 18.

GÉRON, A. *Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow*. [S.l.]: Alta Books, 2019. Citado na página 67.

Goldschmidt, R.; Passos, E. *Data mining: um guia Prático*. Elsevier Editora, 2005. ISBN 978-85-352-1877-0. Disponível em: <https://books.google.com.br/books?id=JJYHNrREwyEC>. Citado 5 vezes nas páginas 32, 33, 34, 35 e 53.

GRUS, J.; NASCIMENTO, W. *Data Science Do Zero*. ALTA BOOKS, 2016. ISBN 9788576089988. Disponível em: <https://books.google.com.br/books?id=EWJCvgAACAAJ>. Citado 2 vezes nas páginas 37 e 38.

GUERRERO-HIGUERAS, A. M.; DeCastro-GARCIA, N.; MATELLAN, V. Detection of cyber-attacks to indoor real time localization systems for autonomous robots. v. 99, p. 75–83, 2018. ISSN 0921-8890. Disponível em: <http://www.sciencedirect.com/science/article/pii/S092188901730283X>. Citado 2 vezes nas páginas 15 e 56.

GUFOSOWA. *K-fold cross validation*. Wikipedia. Disponível em: https://en.wikipedia.org/wiki/File:K-fold_cross_validation_EN.svg, 2019. Disponível em: <https://en.wikipedia.org/wiki/File:K-fold_cross_validation_EN.svg>. Citado na página 39.

HAGAN, M. T.; DEMUTH, H. B.; BEALE, M. H. *Neural Network Design*. [S.l.]: Brooks/Cole, 1996. Google-Books-ID: cUNJAAAACAAJ. ISBN 978-0-534-95259-4. Citado 5 vezes nas páginas 47, 48, 49, 51 e 66.

HAYKIN, S. *Redes Neurais - 2ed*. Bookman, 2001. ISBN 978-85-7307-718-6. Disponível em: <https://books.google.com.br/books?id=KzP4wAEACAAJ>. Citado 9 vezes nas páginas 44, 47, 48, 49, 50, 51, 52, 53 e 58.

JIN, X.; CHEU, R. L.; SRINIVASAN, D. Development and adaptation of constructive probabilistic neural network in freeway incident detection. *Transportation Research Part C: Emerging Technologies*, v. 10, n. 2, p. 121–147, abr. 2002. ISSN 0968-090X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0968090X01000079>. Citado 4 vezes nas páginas 15, 18, 43 e 57.

KADIMISETTY, A. *TensorFlow — A hands-on approach*. [S.l.]: Acesso em 07 de julho de 2020, 2018. (Disponível em: https://towardsdatascience.com/tensorflow-a-hands-on-approach-8614372f021f). Library Catalog: towardsdatascience.com. Citado 2 vezes nas páginas 19 e 67.

KALAM, A. et al. Organization based access control. In: *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*. [S.l.: s.n.], 2003. p. 120–131. Citado na página 27.

KASTURI, S. N. *Underfitting and Overfitting in machine learning and how to deal with it !!!* [S.l.]: Acesso em: 04 de Abril de 2020, 2019. (Disponível em: https://towardsdatascience.com/underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6fe4a8a49dbf). Library Catalog: towardsdatascience.com. Citado na página 37.

KESAVARAJ, G.; SUKUMARAN, S. A study on classification techniques in data mining. In: *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*. [S.l.: s.n.], 2013. p. 1–7. ISSN: null. Citado 3 vezes nas páginas 35, 36 e 40.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. 2017. Disponível em: <http://arxiv.org/abs/1412.6980>. Citado na página 70.

KLUYVER, T. et al. Jupyter notebooks – a publishing format for reproducible computational workflows. In: LOIZIDES, F.; SCHMIDT, B. (Ed.). *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. [S.l.], 2016. p. 87 – 90. Citado na página 62.

KNAPP, K. J. et al. Information security policy: An organizational-level process model. v. 28, n. 7, p. 493–508, 2009. ISSN 0167-4048. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167404809000765>. Citado na página 24.

KOCH, M.; MANCINI, L. V.; PARISI-PRESICCE, F. Conflict Detection and Resolution in Access Control Policy Specifications. In: NIELSEN, M.; ENGBERG, U. (Ed.). *Foundations of Software Science and Computation Structures*. Berlin, Heidelberg: Springer, 2002. (Lecture Notes in Computer Science), p. 223–238. ISBN 978-3-540-45931-6. Citado na página 26.

KROPIWIEC, D. D. Policy viewer: Ferramenta para visualizac¸~ao de pol´ıticas de seguranc¸a em grafos. p. 100, 2005. Citado 2 vezes nas páginas 24 e 25.

KUMAR, R. Classification Algorithms for Data Mining: A Survey. *International Journal of Innovations in Engineering and Technology*, v. 1, n. 2, p. 8, 2012. Citado 2 vezes nas páginas 36 e 40.

LI, N.; TRIPUNITARA, M. V. Security analysis in role-based access control. ACM Trans. Inf. Syst. Secur. ACM, New York. v.9. p.391-420, p. 28, 2006. Citado na página 16.

LIMA, I.; PINHEIRO, C.; SANTOS, F. *Inteligência Artificial*. Elsevier Brasil, 2016. ISBN 978-85-352-7809-5. Disponível em: <https://books.google.com.br/books?id=qjJeBgAAQBAJ>. Citado 7 vezes nas páginas 33, 43, 44, 45, 47, 50 e 58.

LIMA, R. A. F.; PEREIRA, A. C. M. Fraud detection in web transactions. ACM Press, p. 273, 2012. Disponível em: <http://dl.acm.org/citation.cfm?doid=2382636.2382695>. Citado 2 vezes nas páginas 16 e 23.

LOPES, I. M. Adopção de políticas de segurança de sistemas de informação na administração pública local em portugal. 2012. Accepted: 2012-09-03T09:19:35Z Publisher: Universidade do Minho. Disponível em: <https://bibliotecadigital.ipb.pt/handle/10198/7422>. Citado na página 25.

LUGER, G. F. *Inteligência Artificial*. Edição: 4. [S.l.]: ARTMED Editora S.A., 2015. Citado 2 vezes nas páginas 34 e 37.

LUPU, E.; SLOMAN, M. Conflicts in policy-based distributed systems management. *IEEE Transactions on Software Engineering*, v. 25, n. 6, p. 852–869, nov. 1999. ISSN 2326-3881. Citado 4 vezes nas páginas 15, 26, 28 e 55.

MACHADO, A. *Estudo da EMC prevê que volume de dados virtuais armazenados será seis vezes maior em 2020*. 2014. Disponível em: <https://glo.bo/2Eo8AGK>. Citado 2 vezes nas páginas 16 e 23.

MARCIANO, J. L. P. Segurança da informação - uma abordagem social. p. 212, 2006. Citado na página 23.

MCCULLOCH, W. S.; PITTS, W. A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY. v. 52, n. 1, p. 17, 1943. Citado 4 vezes nas páginas 43, 44, 45 e 49.

MCKINNEY Wes. Data Structures for Statistical Computing in Python. In: WALT Stéfan van der; MILLMAN Jarrod (Ed.). *Proceedings of the 9th Python in Science Conference*. [S.l.: s.n.], 2010. p. 56 – 61. Citado na página 62.

MINEWISKAN, O. D. *Modelos de mineração (Analysis Services-Mineração de dados)*. Acesso em: 06 de março de 2020., 2020. (disponível em: https://docs.microsoft.com/pt-br/analysis-services/data-mining/mining-models-analysis-services-data-mining). Library Catalog: docs.microsoft.com. Disponível em: <https://docs.microsoft.com/pt-br/analysis-services/data-mining/mining-models-analysis-services-data-mining>. Citado na página 32.

MINSKY, M.; PAPERT, S. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969. Disponível em: <https://books.google.com.br/books?id=KhI-uwEACAAJ>. Citado 4 vezes nas páginas 43, 44, 48 e 49.

MOFFETT, J. D.; SLOMAN, M. S. Policy conflict analysis in distributed system management. v. 4, n. 1, p. 1–22, 1994. ISSN 1054-1721. Disponível em: <https://doi.org/10.1080/10919399409540214>. Citado na página 26.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. *Foundations of Machine Learning, Second Edition*. 2ª. ed. The MIT Press, 2018. (Adaptive Computation and Machine Learning). ISBN 978-0-262-03940-6. Disponível em: <https://mitpress.mit.edu/books/foundations-machine-learning-second-edition>. Citado na página 33.

MONTEIRO, J. R. Adoção de políticas de segurança de sistemas de informação nas universidades moçambicanas. p. 66, 2017. Citado na página 25.

MUKKAMALA, S.; JANOSKI, G.; SUNG, A. Intrusion detection using neural networks and support vector machines. In: *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*. [S.l.: s.n.], 2002. v. 2, p. 1702–1707 vol.2. ISSN: 1098-7576. Citado 5 vezes nas páginas 15, 24, 54, 56 e 57.

OBAIDAT, M.; MACCHAIROLO, D. A multilayer neural network system for computer access security. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 24, n. 5, p. 806–813, maio 1994. ISSN 2168-2909. Citado 4 vezes nas páginas 15, 18, 43 e 56.

OLIPHANT, T. E. *A guide to NumPy*. [S.l.]: Trelgol Publishing USA, 2006. v. 1. Citado na página 62.

PASZKE, A. et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates, Inc., 2019. 8024–8035 p. Disponível em: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>. Citado na página 19.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado na página 62.

REED, R.; MARKSII, R. J. *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. MIT Press, 1999. Library Catalog: mitpress.mit.edu Publisher: The MIT Press. Disponível em: <https://mitpress.mit.edu/books/neural-smithing>. Citado na página 40.

ROCHA, T. et al. *Tutorial sobre Fuzzy-c-Means e Fuzzy Learning Vector Quantization: abordagens híbridas para tarefas de agrupamento e classificação*. [S.l.: s.n.], 2012. Citado na página 35.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. v. 65, n. 6, p. 386–408, 1958. ISSN 1939-1471(Electronic),0033-295X(Print). Citado 2 vezes nas páginas 43 e 49.

RUSSELL, S.; NORVIG, P. *Inteligêngia Artificial - Tradução da3ª Edição*. Rio de Janeiro, Brasil: Elsevier, 2013. ISBN 9788535237016. Citado 8 vezes nas páginas 34, 36, 37, 38, 39, 43, 44 e 47.

RUUSKA, S. et al. Evaluation of the confusion matrix method in the validation of an automated system for measuring feeding behaviour of cattle. v. 148, p. 56–62, 2018. ISSN 0376-6357. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0376635717301146>. Citado na página 41.

SANDHU, R.; SAMARATI, P. Authentication, access control, and audit. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 28, n. 1, p. 241–243, mar 1996. ISSN 0360-0300. Citado 2 vezes nas páginas 24 e 25.

SANTOS, A. M. R. C. d. Segurança nos sistemas de informação hospitalares : políticas, práticas e avaliação. 2007. Accepted: 2007-12-18T21:06:45Z. Disponível em: <http://repositorium.sdum.uminho.pt/>. Citado na página 23.

SANTOS, F. J. d. *Um Modelo Preditivo de Classificação de Operações de Crédito*. phdthesis — Instituto Federal de Educação do Ceará - IFCE, 2013. Citado na página 43.

SANTOS, T. G. S. *Google Colab: o que é e como usar? | Alura Cursos Online*. [S.l.]: Acesso em 07 de junho de 2020, 2020. (Disponível em: https://www.alura.com.br/artigos/google-colab-o-que-e-e-como-usar). Library Catalog: www.alura.com.br. Citado na página 68.

SARKAR, D.; BALI, R.; SHARMA, T. *Practical Machine Learning with Python: A Problem-Solver's Guide to Building Real-World Intelligent Systems*. Apress, 2017. ISBN 978-1-4842-3207-1. Disponível em: <https://books.google.com.br/books?id=9ClEDwAAQBAJ>. Citado 3 vezes nas páginas 26, 33 e 63.

SARKIS, L. C. *Uma abordagem para detecção de conflitos indiretos entre políticas de controle de acesso*. Tese de Doutorado — Universidade Federal Fluminense, 2017. Citado 12 vezes nas páginas 15, 16, 17, 21, 25, 26, 27, 28, 29, 55, 58 e 62.

SARKIS, L. C.; SILVA, V. T. da; BRAGA, C. Detecting indirect conflicts between access control policies. In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2016. (SAC '16), p. 1570–1572. ISBN 978-1-4503-3739-7. Disponível em: <http://doi.acm.org/10.1145/2851613.2851979>. Citado 6 vezes nas páginas 15, 17, 21, 24, 27 e 55.

SHAZMEEN, S. F.; MUSTAFA, M. A.; BAIG, A. Performance Evaluation of Different Data Mining Classification Algorithm and Predictive Analysis. In: . [S.l.: s.n.], 2013. Citado 2 vezes nas páginas 35 e 36.

SHOHAM, Y.; TENNENHOLTZ, M. On social laws for artificial agent societies: off-line design. v. 73, n. 1, p. 231–252, 1995. ISSN 00043702. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/000437029400007N>. Citado 2 vezes nas páginas 18 e 73.

SILVA, I. N. d. *Redes Neurais Artificiais Para Engenharia e Ciências Aplicadas. Fundamentos Teóricos e Aspectos Práticos*. Edição: 2ª. [S.l.]: Artliber, 2016. ISBN 978-85-88098-87-9. Citado 4 vezes nas páginas 46, 47, 48 e 58.

SILVA, L. A. d.; PERES, S. M.; BOSCARIOLI, C. *Introdução à Mineração de Dados: Com Aplicações em R*. Elsevier Brasil, 2017. ISBN 9788535284478. Disponível em: <https://books.google.com.br/books?id=5LA4DwAAQBAJ>. Citado 13 vezes nas páginas 16, 23, 30, 31, 32, 34, 35, 37, 38, 39, 40, 41 e 42.

SILVA, L. C. e. *Aprendizado de máquina com treinamento continuado aplicado à previsão de demanda de curto prazo: o caso do restaurante universitário da Universidade Federal de Uberlandia*. Mestrado em Engenharia Elétrica — Universidade Federal de Uberlandia, 2019. Citado na página 34.

SILVESTRE, E. A. *Verificação de conflitos entre múltiplas normas em sistemas multiagentes*. Tese de Doutorado — UFF - Universidade Federal Fluminense, 2017. Citado 4 vezes nas páginas 15, 17, 18 e 55.

SIMON, P. *Too Big to Ignore: The Business Case for Big Data*. [S.l.]: Wiley, 2013. (Wiley and SAS Business Series). ISBN 978-1-118-64186-6. Citado na página 33.

SLOMAN, M.; LUPU, E. Security and management policy specification. v. 16, n. 2, p. 10–19, 2002. ISSN 08908044. Disponível em: <http://ieeexplore.ieee.org/document/993218/>. Citado 2 vezes nas páginas 26 e 28.

SWAMYNATHAN, M. *Mastering Machine Learning with Python in Six Steps: A Practical Implementation Guide to Predictive Data Analytics Using Python*. 2ª. ed. Apress, 2019. ISBN 978-1-4842-4947-5. Disponível em: <https://books.google.com.br/books?id=dEqzDwAAQBAJ>. Citado na página 33.

THENMOZHI, D.; KALAIVANI, A.; ARAVINDAN, C. Multi-lingual author profiling on SMS messages using machine learning approach with statistical feature selection. p. 9, 2018. Disponível em: <https://pdfs.semanticscholar.org/5d3e/7741c33bc7613fd4d5a7711d7f33830dc28f.pdf>. Citado na página 68.

UEDA, E. T. *Análise de políticas de controle de acesso baseado em papéis com rede de Petri colorida*. Doutorado em Sistemas Digitais — USP - Universidade de São Paulo, 2012. Disponível em: <http://www.teses.usp.br/teses/disponiveis/3/3141/tde-08032013-120904/>. Citado na página 16.

VASCONCELOS, L. M. R. d.; CARVALHO, C. L. d. Aplicação de regras de associação para mineração de dados na web. v. 1, n. 1, 2018. Number: 1. Disponível em: <https://telematicafractal.com.br/revista/index.php/telfract/article/view/8>. Citado na página 31.

VIMERCATI, S. De Capitani di; SAMARATI, P.; JAJODIA, S. Policies, models, and languages for access control. In: BHALLA, S. (Ed.). *Databases in Networked Information Systems*. [S.l.]: Springer, 2005. (Lecture Notes in Computer Science), p. 225–237. ISBN 978-3-540-31970-2. Citado 3 vezes nas páginas 25, 26 e 27.

WANG, Y. et al. Conflicts analysis and resolution for access control policies. In: *2010 IEEE International Conference on Information Theory and Information Security*. [S.l.: s.n.], 2010. p. 264–267. ISSN: null. Citado 2 vezes nas páginas 24 e 25.

WIEDERHOLD, G.; McCarthy, J. Arthur samuel: Pioneer in machine learning. v. 36, n. 3, p. 329–331, 1992. ISSN 0018-8646. Citado na página 33.

WITTEN, I. et al. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier Science, 2016. (The Morgan Kaufmann Series in Data Management Systems). ISBN 978-0-12-804357-8. Disponível em: <https://books.google.com.br/books?id=1SylCgAAQBAJ>. Citado 3 vezes nas páginas 59, 60 e 76.

WOLPERT, D. H. The lack of a priori distinctions between learning algorithms. *Neural Comput.*, MIT Press, Cambridge, MA, USA, v. 8, n. 7, p. 1341–1390, oct 1996. ISSN 0899-7667. Disponível em: <http://dx.doi.org/10.1162/neco.1996.8.7.1341>. Citado 2 vezes nas páginas 36 e 39.

WOLPERT, D. H.; MACREADY, W. G. Working Papers. *No Free Lunch Theorems for Search*. Santa Fe Institute, 1995. Disponível em: <https://EconPapers.repec.org/RePEc:wop:safiwp:95-02-010>. Citado na página 39.

WU, X. et al. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, Springer-Verlag New York, Inc., New York, NY, USA, v. 14, n. 1, p. 1–37, dec 2007. ISSN 0219-1377. Disponível em: <http://dx.doi.org/10.1007/s10115-007-0114-2>. Citado na página 60.

YEUNG, N.; BOTVINICK, M. M.; COHEN, J. D. The neural basis of error detection: conflict monitoring and the error-related negativity. *Psychological Review*, v. 111, n. 4, p. 931–959, out. 2004. ISSN 0033-295X. Citado na página 48.