Luis Henrique Meazzini Sepulvene

Aplicação de técnicas de aprendizado de máquina para o diagnóstico de falhas em módulos rastreadores de frotas veiculares

Itajubá

Maio de 2019

Luis Henrique Meazzini Sepulvene

Aplicação de técnicas de aprendizado de máquina para o diagnóstico de falhas em módulos rastreadores de frotas veiculares

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciência e Tecnologia da Computação.

Universidade Federal de Itajubá – UNIFEI

Instituto de Engenharia de Sistemas e Tecnologia da Informação – IESTI

Mestrado em Ciência e Tecnologia da Computação

Orientador: Bruno Guazzelli Batista

Coorientador: Isabela Neves Drummond

Itajubá

Maio de 2019

Agradecimentos

Dedico este trabalho primeiramente a Deus, criador do céu e da terra, de todas as coisas visíveis e invisíveis. Também, agradeço aos meus pais Edwaguinei e Cláudia por todo suporte, incentivo e apoio incondicional. Agradeço também ao meu irmão Pedro, pela sua alegria e seus dotes culinários. Juntamente, devo meus sinceros agradecimentos à flor que dá doçura a vida, que me inspira e guia, Larissa Adriene Ramos Ferreira.

A meu orientador, Bruno Guazzelli Batista e a minha co-orientadora Isabela Neves Drummond pela atenção, dedicação e suporte durante todo este período. Também agradeço a DDMX pela disponibilidade e incentivo em diversas partes desta pesquisa, ademais ao grupo de pesquisa DDMX-Sinape, especialmente ao professor Rafael de Magalhães Dias Frinhani pela diversas reuniões e direcionamentos no grupo.

Outrossim, ao grande amigo Gabriel de Souza Pereira Gomes conjuntamente com Letícia Maria Peres Gomes e Helena Maria Peres Gomes, pela instigação à pesquisa e desenvolvimento em toda engenharia num contexto geral. Além do mais, a outros amigos especiais como: Pablo José Freitas de Souza, Yuri Miranda Santos, Guilherme Augusto Carvalho Camanducaia, Danilo Ishuah Mendonça Viana, Rafael Henrique Rodrigues, Victor Rios Belarmino, Márcio Aduil Gomes e João Pedro Serpa Dias.

"Há apenas uma maneira de evitar críticas:
não falar, não fazer e não ser nada."
(Aristóteles)

Resumo

Com a indústria 4.0, as abordagens baseadas em dados estão em voga. No entanto, extrair as características importantes não é uma tarefa fácil e influencia muito o resultado final. Também há a necessidade de um conhecimento especializado do sistema para monitorar o ambiente e diagnosticar falhas. Neste contexto, o diagnóstico de falhas é significativo, por exemplo, em um sistema de monitoramento de frotas de veículos, pois é possível diagnosticar falhas antes mesmo que um cliente saiba da existência desta falha, além de minimizar os custos de manutenção dos módulos. Neste trabalho são propostas duas abordagens, "com informação" e "sem informação", para exploração de um conjunto de dados, empregando técnicas de Aprendizado de Máquina (AM) para geração de modelos classificadores que auxiliem no processo de diagnóstico de falhas em módulos rastreadores de frotas veiculares. A abordagem "com informação" realiza a extração de características de forma manual, empregando os modelos de AM: *Random Forest*, *Naive Bayes*, Máquina de vetor de suporte (SVM) e Perceptron de múltiplas camadas (MLP); e a abordagem "sem informação" realiza a extração de características de forma automática, através de uma rede neural convolucional (CNN). Os resultados obtidos demonstraram que as abordagens propostas são promissoras. Os melhores modelos com extração de características manual obtiveram uma precisão de 99,76% e 99,68% para detecção e detecção e identificação de falhas, respectivamente, no conjunto de dados fornecido. Os melhores modelos fazendo uma extração de características automática obtiveram respectivamente 88,43% e 54,98% para detecção e detecção e identificação de falhas. Estes modelos podem servir como protótipos para diagnosticar falhas remotamente e confirmam que as técnicas tradicionais de AM com uma extração de características manual ainda são recursos eficazes para o diagnóstico de falhas.

Palavras-chave: Diagnóstico de falhas; Módulos rastreadores; Aprendizado de máquina; Extração de características.

Abstract

With industry 4.0, data-based approaches are in vogue. However, extracting the essential features is not an easy task and greatly influences the final result. There is also a need for specialized system knowledge to monitor the environment and diagnose faults. In this context, the diagnosis of faults is significant, for example, in a vehicle fleet monitoring system, since it is possible to diagnose faults even before the customer is aware of the fault, in addition to minimizing the maintenance costs of the modules. In this work, several models using Machine Learning (ML) techniques were applied and analyzed during the fault diagnosis process in vehicle fleet tracking modules. This research proposes two approaches, with knowledge and without knowledge, to explore the dataset using ML techniques to generate classifiers that can assist in the fault diagnosis process in vehicle fleet tracking modules. The approach with knowledge performs the feature extraction manually, using the ML techniques: Random Forest, Naive Bayes, SVM and MLP; and the approach without knowledge performs an automatic feature extraction, through a Convolutional Neural Network (CNN). The results showed that the proposed approaches are promising. The best models with manual feature extraction obtained a precision of 99,76% and 99,68% for detection and detection and identification of faults, respectively, in the provided dataset. The best models performing an automatic feature extraction obtained respectively 88,43% and 54,98% for detection and detection and identification of failures. These models can serve as prototypes to diagnose faults remotely and confirm that traditional ML techniques with manual extraction of features are still effective resources for fault diagnosis.

Keywords: Fault Diagnosis; Tracker modules; Machine learning; Feature extraction.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AM	Aprendizado de Máquina
BD	Banco de Dados
CNN	*Convolutional Neural Network*
DL	*Deep Learning*
DSM	*Deep & Shallow Method*
FDI	*Fault Detection and Isolation*
FFT	*Fast Fourier Transform*
FL	*Fuzzy Logic*
HDN	*Hierarchical Diagnosis Network*
LSTM	*Long-Short Term Memory*
MLP	*Multi Layer Perceptron*
PCA	*Principal Component Analysis*
PLS	*Partial Least Squares*
RBF	*Radial Basis Function*
RUS	*Random Undersampling*
ReLU	*Rectified Linear Unit*
RNA	Rede Neural Artificial
RNN	Rede Neural Recorrente
SGD	*Stocastic Gradient Descent*
SVM	*Support Vector Machine*
TVP	Taxa de Verdadeiros Positivos
WPT	*Wavelet Packet Transform*

Sumário

1 Introdução

O diagnóstico de falhas tem sua importância devido à demanda de confiabilidade em sistemas sujeitos a possíveis anormalidades e falhas de componentes, sendo que estas falhas podem resultar em um mau funcionamento ou parada do sistema. Desta forma, é fundamental detectar e identificar possíveis anomalias e falhas o mais cedo possível a fim de minimizar a degradação do sistema, evitar situações perigosas e prejuízos financeiros.

Existem diversas abordagens de diagnóstico de falhas, como as baseadas em modelos, sinais e conhecimento (GAO; CECATI; DING, 2015a). No entanto, com a grande quantidade de dados existentes, os métodos de diagnóstico de falhas estão fazendo o uso destes dados para detectar, isolar e identificar falhas em sistemas. Assim, nota-se que as abordagens tradicionais de diagnóstico de falhas necessitam de um modelo matemático do sistema, o que é dispensável quando se usa uma técnica de aprendizado de máquina (AM) para o diagnóstico (STOJANOVIC et al., 2016).

Certamente, um dos maiores desafios no diagnóstico de falhas usando métodos de AM é que os dados devem ser analisados com cautela e devidamente tratados a fim de que o método possa convergir e obter um bom resultado. Khan e Yairi (2018) listam alguns desafios desta abordagem como a definição de um único procedimento de diagnóstico, mapeamento robusto das falhas, sensores com ruídos, dados faltando, interdependência dos processos do sistema e falta de conhecimento especializado.

Abordagens tradicionais baseadas em conhecimento geralmente possuem uma fase de extração de características, na qual existe uma demanda por conhecimento especializado do sistema (ZHAO et al., 2019). Esta etapa consiste na seleção dos atributos mais representativos e na criação de novos atributos artesanais (MALHI; GAO, 2004). No entanto, existem atualmente métodos que não extraem características e trabalham com todos os dados possíveis, como o *Deep Learning* (DL) (ZHAO et al., 2019).

Muitas empresas de transporte e logística fazem uso de módulos para rastrear seus veículos, nos quais é comum descobrirem que alguns módulos não estão funcionando como esperado. Assim, é significativo fazer um diagnóstico remoto de falhas destes módulos, uma vez que, na ocorrência de falhas, perdas financeiras podem ocorrer. O veículo talvez tenha que ficar parado, dados errados do veículo podem ser coletados e também um suporte técnico pode ser enviado para analisar o problema em uma longa distância, por exemplo.

1.1 Motivação

Esta pesquisa é motivada pelo fato de que o diagnóstico de falhas é uma tarefa de grande importância quando se deseja maximizar o aproveitamento do sistema projetado. A partir desta afirmação, combinada com a quantidade de dados disponíveis e a dificuldade de modelar matematicamente determinados sistemas, as abordagens de diagnóstico de falhas baseadas em dados tem sua relevância cada vez maior.

Enquanto abordagens tradicionais de AM como Árvores de decisão, *Naive Bayes* e SVM não trabalham bem com uma grande quantidades de dados, estas necessitam de uma extração de características que seja eficiente para apresentar resultados precisos (LIU; MOTODA, 2012). Enquanto isso, as abordagens utilizando DL são capazes de trabalhar com grandes quantidades de dados e não fazem necessária a extração de características, podendo assim extrair as características relevantes do sistema de forma automática (LIU et al., 2016).

Desta forma, um fator motivacional é analisar e comparar quantitativamente a influência de uma boa extração de características em relação a uma extração de características automática através de um método inteligente. Outros fatores motivacionais são a minimização dos gastos com suporte técnico em módulos rastreadores de frotas veiculares e a utilização de dados reais oriundos destes módulos para o diagnóstico de falhas, já que não foram encontrados trabalhos considerando estes dados.

Outro fator importante é que as técnicas de aprendizado de máquina podem ser utilizadas em diversas áreas além do diagnóstico de falhas, como diagnóstico médico (KONONENKO, 2001), genética (LIBBRECHT; NOBLE, 2015), robótica e diversas outras áreas (NASRABADI, 2007). Assim, analisar aplicações destas técnicas podem render frutos para as demais áreas de aplicação do aprendizado de máquina.

1.2 Objetivo

Em face ao exposto, o objetivo deste trabalho consiste em aplicar técnicas de aprendizado de máquina para diagnosticar falhas em módulos rastreadores de frotas veiculares. Para isso, serão considerados dados reais coletados destes módulos.

A partir dessa asserção, os objetivos específicos desta pesquisa são:

- Construir modelos capazes de diagnosticar falhas no sistema proposto, usando conhecimento de um especialista do sistema para realizar uma extração de características;
- Desenvolver modelos para o diagnóstico de falhas no sistema proposto, sem usar qualquer tipo de conhecimento prévio do sistema e apenas com dados crus recebidos

dos módulos;

- Comparar e investigar a eficiência de cada abordagem e cada modelo para diagnosticar falhas;

As principais contribuições desta pesquisa são: i) a aplicação de uma metodologia para diagnóstico de falhas em módulos de rastreamento de frotas veiculares que pode ser usada para extrair características de sistemas similares; ii) modelos capazes de detectar e identificar falhas em dados oriundos de módulos rastreadores de frotas veiculares e; iii) uma análise comparativa entre todas abordagens e métodos utilizados na construção dos modelos em questão.

1.3 Estrutura da dissertação

Este documento foi estruturado em cinco capítulos, da seguinte forma:

- **Capítulo 1** - apresenta a introdução e a justificativa para o desenvolvimento do trabalho, seus objetivos e sua organização;
- **Capítulo 2** - disserta sobre a revisão bibliográfica no que se refere as abordagens mais comuns de diagnóstico de falhas;
- **Capítulo 3** - disserta sobre a revisão bibliográfica no que se refere as abordagens do aprendizado de máquina e os trabalhos correlacionados, expondo os trabalhos atuais relacionados ao estudo de caso desta pesquisa;
- **Capítulo 4** - apresenta o problema tratado por esta pesquisa e as abordagens propostas para o diagnóstico de falhas;
- **Capítulo 5** - apresenta o planejamento e aplicação dos experimentos, e por fim, a discussão dos resultados obtidos;
- **Capítulo 6** - expõe as conclusões desta pesquisa e ideias para trabalhos futuros.

Após a conclusão e trabalhos futuros, são apresentadas as referências bibliográficas que embasaram esta pesquisa.

2 Diagnóstico de falhas em Sistemas de Informação

Muitos sistemas de engenharia como aeromotores, processos químicos, redes de energia, equipamentos industriais eletrônicos, entre outros, são sensíveis a falhas. Assim existe uma demanda cada vez maior de confiabilidade e segurança já que estes sistemas estão sujeitos a anormalidades de processo e falhas de componentes. Consequentemente, é fundamental detectar e identificar qualquer tipo de anormalidade e falha, o mais cedo possível, para que estas sejam corrigidas ou prevenidas com maior eficiência, evitando situações perigosas e perdas financeiras (GAO; CECATI; DING, 2015a).

Pode-se dizer que um sistema confiável é um sistema que não apresenta falhas. Dessa forma, devido às crescentes demandas por um maior desempenho e qualidade de produto, a complexidade e grau de automação dos processos técnicos estão crescendo continuamente. Atualmente, uma das questões mais críticas em torno do projeto de um sistema automático é a confiabilidade do sistema (DING, 2008).

A partir da década de 1970, o diagnóstico de falhas tornou-se uma área em constante crescimento. No entanto, os métodos usados nem sempre foram os mesmos. A princípio, as metodologias baseadas em modelos eram mais comuns. A partir da década de 1990, as demais metodologias, citadas nas próximas seções, ganharam espaço graças aos avanços tecnológicos (WILLSKY, 1976; SOTTILE; HOLLOWAY, 1994).

Jia et al. (2018) dizem que atualmente o diagnóstico de falhas inteligente é um das ferramentas mais poderosas. Além disto, este diagnóstico geralmente é feito baseado em dados, fazendo o uso de grandes bancos de dados para treinar algoritmos de aprendizado de máquina.

Diferentes pontos de vista podem ser escolhidos para a implementação de um algoritmo de diagnóstico de falhas. Esta escolha geralmente é baseada no nível de conhecimento que pode ser retirado do sistema (STAROSWIECKI, 2001). Desta forma, o diagnóstico de falhas baseado na análise de dados tem uma grande importância em diversos sistemas.

2.1 Diagnóstico de falhas

Schrick (1997) definem falha como um desvio não permitido de pelo menos uma propriedade ou parâmetro característico do sistema. Segundo (CHEN; PATTON, 2012), o processo de diagnóstico de falhas inclui três tarefas:

- **Detecção de falhas**: é a tarefa mais básica do diagnóstico de falhas, serve para verificar se há mau funcionamento ou falha no sistema e determinar em que momento a falha ocorre;
- **Isolamento de falhas**: o isolamento serve para detectar a localização do componente defeituoso;
- **Identificação de falhas**: a identificação serve para determinar o tipo, forma e o tamanho da falha.

Geralmente, o processo consiste apenas na detecção e isolamento da falha, ou *Fault Detection and Isolation* (FDI). Isto não nega a utilidade da identificação de falhas, mas este processo pode não ser essencial se nenhuma ação de reconfiguração estiver envolvida (CHEN; PATTON, 2012).

Durante as últimas quatro décadas, vários métodos e avanços na área de diagnóstico de falhas foram feitos (WILLSKY, 1976; GERTLER, 1988; FRANK; DING, 1997; VENKATASUBRAMANIAN et al., 2003; YIN et al., 2014; GAO; CECATI; DING, 2015a; GAO; CECATI; DING, 2015b). Algumas metodologias são descritas nas próximas seções.

2.2 Métodos baseados em modelos

Originado em Beard (1971), com a ideia de substituir a redundância de *hardware* por redundância analítica. Neste tipo de abordagem é necessário que os modelos dos processos estejam disponíveis, os quais podem ser obtidos aplicando princípios físicos e técnicas de modelagem de sistemas. Estes algoritmos de diagnóstico de falhas tem como objetivo monitorar a consistência entre as saídas medidas e as saídas esperadas pelo modelo.

Nas abordagens não baseadas em modelos, os dados históricos anteriores são analisados a fim de detectar regularidades e padrões que ligariam os dados às conclusões finais. Já as técnicas baseadas em modelos, usam um conhecimento sobre o comportamento comum do sistema e analisam os dados emitidos pelo sistema para detectar divergências entre o que deveria ser a saída. Nota-se a necessidade de um modelo matemático do sistema para que seja possível realizar um diagnóstico de falhas baseado em modelo (STAROSWIECKI, 2001).

Os métodos baseados em modelos fazem uso de uma redundância analítica como pode ser visto na Figura 1. A ideia destes modelos reside em comparar a saída esperada com a saída real do processo, e para isto é necessário um modelo matemático do sistema, ou seja, é necessário que seja possível fazer um modelo matemático analítico que represente este sistema. Assim, na Figura 1 pode-se notar que a abordagem baseada em modelos visa

criar uma redundância analítica com um algoritmo de FDI para substituir a redundância de *hardware*, que pode ter um custo elevado.

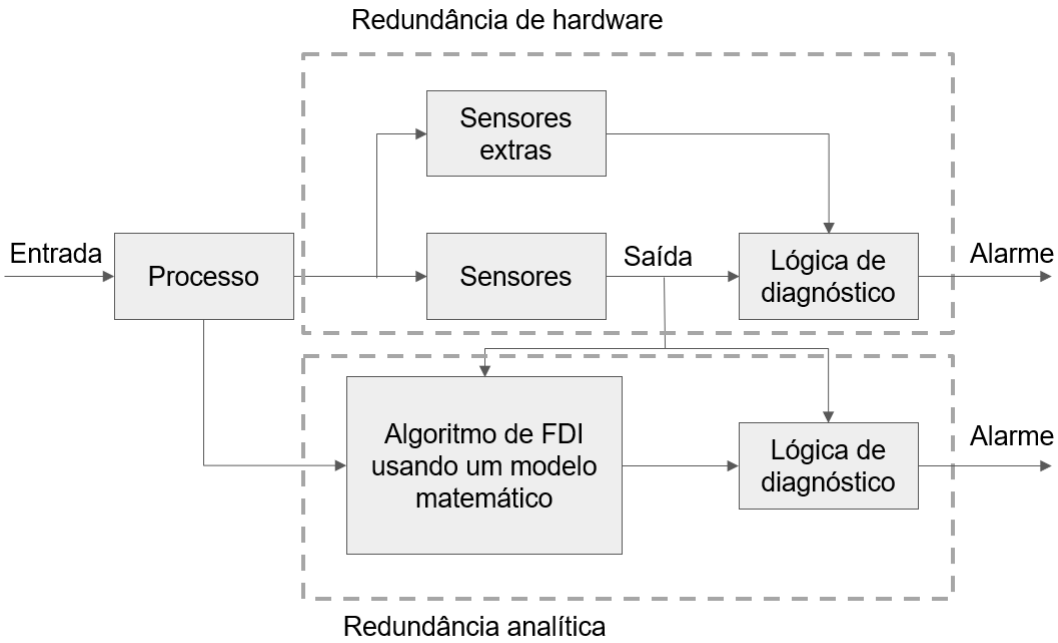


Figura 1 – Funcionamento dos métodos baseados em modelos, adaptado de (CHEN; PATTON, 2012)

Garcia e Frank (1997) explicam como os métodos baseados em modelos detectam e isolam as falhas. Primeiramente é feita a construção de um gerador residual, que geralmente é a diferença entre a saída esperada e a saída real do sistema. Em um caso ideal o gerador residual será nulo. Em caso de falhas, este gerador residual será diferente de zero e a falha será indicada pela forma do gerador residual.

Os métodos de diagnóstico de falhas baseados de modelos podem ser divididos em quatro categorias (GAO; CECATI; DING, 2015a):

- **Métodos de diagnóstico de falhas determinísticos**: para os métodos determinísticos, um ou mais observadores são usados para a construção do gerador residual;

- **Métodos de diagnóstico de falhas estocásticas**: estes métodos foram desenvolvidos em paralelo com os métodos determinísticos, inicialmente usando filtros de *Kalman* como similares a um observador. Em seguida, várias outras técnicas estatísticas, como testes de χ^2, algoritmos de soma cumulativa e testes de múltiplas hipóteses; também foram utilizados para criar algoritmos de FDI. Além disso, métodos como a estimativa de parâmetros são bem utilizados no diagnóstico de falhas (ISERMANN, 1984; DÖHLER; MEVEL, 2013).

- **Métodos de diagnóstico de falhas para eventos discretos e sistemas híbridos**: alguns processos possuem sinais discretos e não contínuos. Dessa forma, foi necessário desenvolver métodos de diagnóstico de falhas especificamente para este tipo de sistema, o que foi inicialmente feito por Sampath et al. (1995). O desafio do diagnóstico de falhas baseado em eventos é executar uma inferência baseada em modelo em tempo real para determinar se uma falha ocorreu, fazendo o uso de sequências de eventos passados observados. Dessa forma, o diagnóstico de falhas para eventos discretos pode ser dividido em duas categorias:

  - **Redes de Petri:** estas redes tem uma natureza distribuída, onde as noções de estado e ação são locais, o que é útil para reduzir a complexidade computacional. Os bons resultados foram desenvolvidos usando programação linear com redes de Petri ou com redes de Petri parcialmente observadas (DOTOLI et al., 2009; CABASINO; GIUA; SEATZU, 2010).
  - **Baseado em autômatos:** estes métodos podem ser descentralizados, centralizados ou híbridos, sendo que por centralizado entende-se que o autômato diagnosticador tem acesso a todos eventos observáveis do sistema, enquanto que nos métodos descentralizados, a leitura dos sensores é distribuída em diferentes módulos. Os métodos híbridos fazem uma mistura dos centralizados e descentralizados (BASILIO; CARVALHO; MOREIRA, 2010).

  Também existem sistemas híbridos, com processos contínuos e outros discretos. Para este tipo de sistema geralmente são usados métodos baseados em autômatos e grafos de ligação (GUO et al., 2013; LEVY; AROGETI; WANG, 2014).

- **Métodos de diagnóstico de falhas para sistemas distribuídos**: O objetivo deste tipo de método não deve ser apenas contra erros de modelagem, distúrbios de processo, ruídos de medição, mas também contra atrasos de transmissão, perda de dados e medições incompletas. Para isto, uma variedade de técnicas foi desenvolvida para sistemas distribuídos. Alguns exemplos de técnicas usam o filtro de *Kalman*, filtro de quadrados mínimos e modelos generalizados de *Poisson* (LEI; YUAN; ZHAO, 2014; KELKAR; KAMAL, 2014).

2.3 Métodos baseados em sinais

Estes métodos usam sinais medidos em vez de modelos explícitos. As falhas são refletidas nos sinais medidos e a decisão diagnóstica é feita com base na análise dos sinais e no conhecimento prévio sobre o comportamento do sinal. Os métodos baseados em sinais podem fazer o uso de características extraídas, tanto no domínio do tempo quanto no domínio da frequência (GAO; CECATI; DING, 2015a).

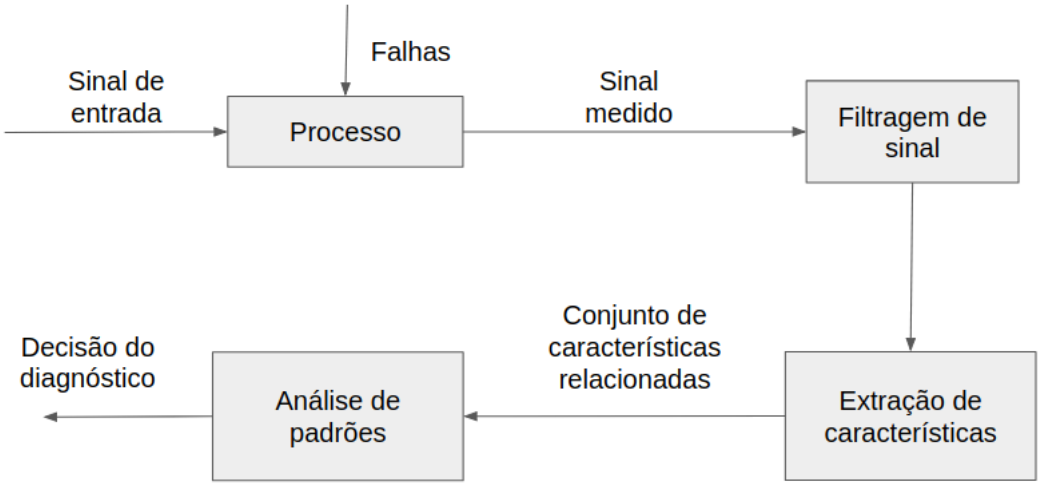


Figura 2 – Funcionamento dos métodos baseados em sinais, adaptado de (OLSSON; FUNK; XIONG, 2004)

Como pode ser visto na Figura 2, um método baseado em sinal possui um sinal de entrada, e um outro sinal de saída. O sinal de saída é filtrado, e depois disso, ocorre uma extração das características do sinal. Com as características do sinal já identificadas, é feita uma análise com um conjunto de características que configura falha, e assim é possível realizar o FDI (OLSSON; FUNK; XIONG, 2004).

As características do sinal de saída são extraídas para a análise de padrões, estas características podem estar no domínio do tempo ou da frequência. Assim, os métodos baseados em sinais, podem ser diferenciados em como são extraídas as características dos sinais (GAO; CECATI; DING, 2015a):

- **Métodos baseados em sinais no domínio do tempo**: na monitorização de um sinal, é natural fazer uma extração de características no domínio do tempo. Para isto, alguns atributos podem ser usados, como o ângulo da fase, amplitude, sequência de zeros e covariância. Além disso, é possível fazer o uso de técnicas de distorção de tempo dinâmico (HONG; DHUPIA, 2014);

- **Métodos baseados em sinais no domínio da frequência**: estes métodos fazem a análise do espectro do sinal usando a transformada rápida de *Fourier Fast Fourier Transform* (FFT), e outros métodos como *wavelets*;

- **Métodos baseados em sinais no domínio do tempo e frequência**: como o espectro de um sinal pode ser variável no tempo, muitas vezes é necessário fazer uma análise do sinal nos domínios do tempo e frequência. Para tal tarefa, vários métodos podem ser utilizados, como a transformada de *Fourier* de curta duração,

distribuição de *Wigner*, transformada de *Wavelet*, transformada de *Hilbert-Huang* e distribuição de *Wigner-Ville* (HLAWATSCH; AUGER, 2013).

2.4 Métodos baseados em conhecimento

Os métodos baseados em conhecimento consistem em usar dados históricos do processo a ser analisado para treinar um algoritmo, o qual será capaz de avaliar os dados atuais do processo para diagnosticar suas falhas (RUSSELL; CHIANG; BRAATZ, 2012).

É importante ressaltar que estes métodos devem utilizar dados em tempo real ao realizar um diagnóstico em tempo real, e isto não remove a necessidade de haver uma grande quantidade de dados históricos do processo.

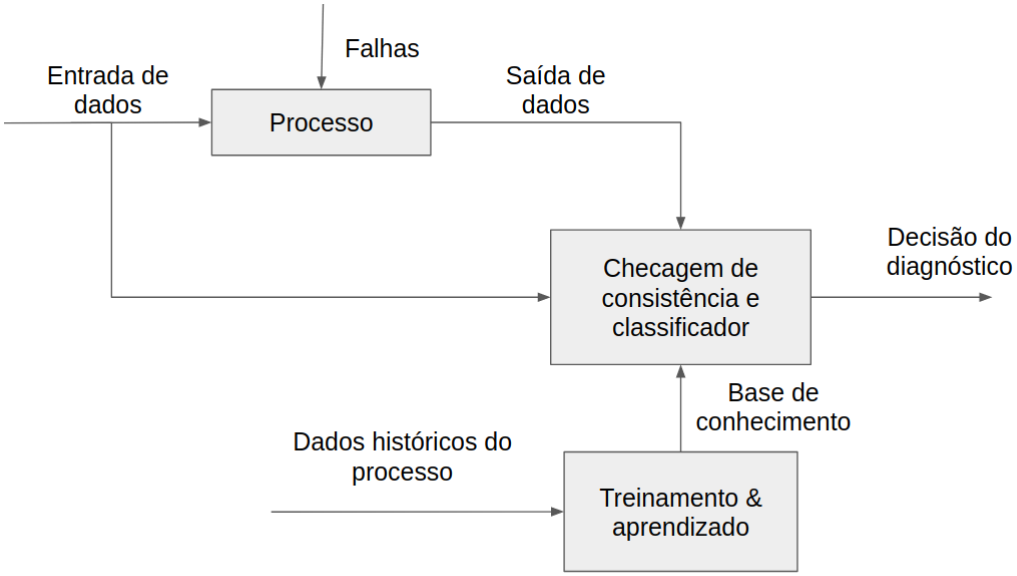


Figura 3 – Esquema de diagnóstico baseado em conhecimento, adaptado de (GAO; CECATI; DING, 2015b)

Na Figura 3, pode-se observar o funcionamento de um método de diagnóstico baseado em conhecimento. Para verificar a consistência de dados, é necessário que a entrada e a saída do processo sejam comparadas com uma base de conhecimento, gerada através de dados históricos. Caso exista algum erro grosseiro ou outro tipo de inconsistência, é possível que se descarte este dado. O FDI também é feito no bloco classificador, relacionando a saída de dados com a base de conhecimento disponível.

As informações da base de dados podem ser qualitativa ou quantitativa. Desta forma o diagnóstico de falhas baseado em conhecimento pode ser quantitativo ou qualitativo.

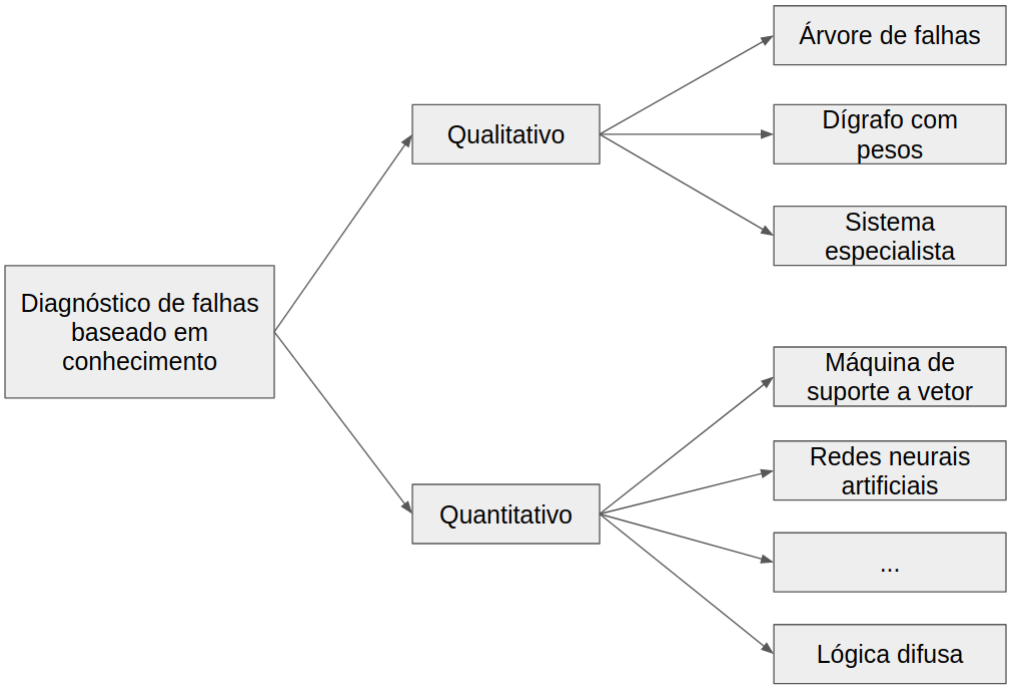


Figura 4 – Métodos de diagnóstico de falhas

Na Figura 4, diversos métodos de diagnóstico de falhas baseados em conhecimento podem ser vistos. Já divididos em qualitativos e quantitativos, estes métodos serão melhor abortados nas subseções posteriores.

2.4.1 Diagnóstico de falhas baseado em conhecimento qualitativo

Os métodos qualitativos são divididos em três sub-categorias: árvore de falhas, grafos dirigidos com peso e sistemas especialistas.

A árvore de falha foi desenvolvida pelo *Bell Lab*, na década de 1960, como uma árvore com uma lógica de causa-efeito, que propaga seus eventos primários (falhas) das folhas para a raiz (sintomas). Nota-se que, para utilizar este método é necessário um conhecimento do funcionamento do processo (RUIJTERS; STOELINGA, 2015). Este funcionamento pode ser visto na Figura 5, na qual pode-se observar que as falhas em uma escada são analisadas, e as ramificações desta árvore ligam causas e efeitos até chegar a uma folha que representa a falha. Por exemplo, caso uma escada esteja instável, notamos que é sua estabilidade que está comprometida e temos as opções de ser uma falha no material da escada ou uma falha em sua construção.

Liu et al. (2014) fizeram uma análise baseada no método da árvore de falhas, para tomada de decisões de risco em situações emergenciais. Para demonstrar a validade do método, foi feito também um estudo de caso com a doença do vírus H1N1.

Um dígrafo com peso pode ser usado para o diagnóstico de falhas, fazendo com

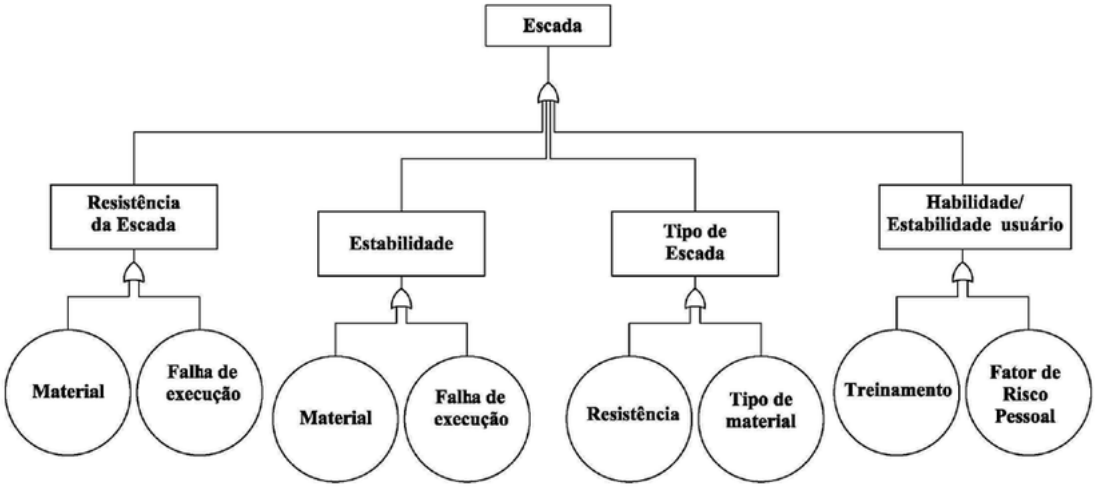


Figura 5 – Exemplo de árvore de falhas (LANA et al., 2014).

que as arestas do grafo sejam de uma causa para um efeito, e seus pesos sejam positivos ou negativos, indicando a correlação de causa e efeito. Liu et al. (2016) definiram um grafo para o diagnóstico de falhas em usinas nucleares. Um exemplo de aresta deste grafo é a ligação com peso positivo entre um nó, representando baixa temperatura da água, e a falha na bomba de circulação de água.

A área de sistemas especialistas surgiu no final da década de 60, como um ramo da inteligência artificial. A partir da década de 80, os sistemas especialistas começaram a ser utilizados para o diagnóstico de falhas. Estes sistemas avaliam regras aprendidas por humanos para classificarem o sistema quanto a sua falha (DAI; GAO, 2013; CHESTER; LAMB; DHURJATI, 1984). Mostafa et al. (2012) implementaram um sistema especialista para assistência de falhas e mau funcionamento de carros, para tal, fizeram o uso de conhecimento especializado sobre o sistema e regras simples para diagnosticar falhas, obtendo um resultado satisfatório.

2.4.2 Diagnóstico de falhas baseado em conhecimento quantitativo

O diagnóstico de falhas baseado em conhecimento quantitativo, consiste em sua essência, em desenvolver uma solução com um problema de reconhecimento de padrões. Os dados quantitativos são analisados usando métodos estatísticos ou não estatísticos (GAO; CECATI; DING, 2015b).

Dai e Gao (2013) afirmam que o aprendizado de máquina é uma forma eficiente de retirar conhecimento de uma grande quantidade de dados empíricos, com o custo relacionado ao processamento massivo de dados. Além disso, tornou-se fácil aplicar técnicas de aprendizado de máquina para detectar e diagnosticar falhas de dados, sem a necessidade de um modelo explícito.

Dentro da análise estatística, Yin et al. (2015) relatam que as ferramentas mais utilizadas são a análise de componentes principais ou *Principal Component Analysis* (PCA), e os mínimos quadrados parciais ou *Partial Least Squares* (PLS):

- ***Principal Component Analysis* - PCA**: é a técnica mais utilizada de monitoramento baseada em estatística. Ela é utilizada para encontrar fatores com uma dimensão menor que o conjunto de dados original, de forma que esta técnica usa dados representados por uma matriz de x registros por y atributos, que podem estar correlacionados, sumarizando este conjunto por eixos não correlacionados, que são uma combinação linear das y variáveis originais a fim de obter uma matriz menor.

- ***Partial Least Squares* - PLS**: é um método dominante do diagnóstico de falhas em processos industriais complexos. Este método é usado para descrever uma relação entre um conjunto preditivo, e uma ou mais respostas do sistema. Assim, o PLS pode detectar padrões de resposta e relações entre os dados de entrada e saída.

Além disso, Yin et al. (2015) afirmam que a análise estatística é uma ferramenta eficaz para lidar com uma grande quantidade de dados. Vários estudos usando métodos de diagnóstico de falhas fazem o uso do PCA, e estes tem bons resultados. Um exemplo disso é o trabalho desenvolvido por Wang et al. (2008), no qual é apresentado um estudo que utiliza uma extensão não linear do PCA juntamente com uma MLP para diagnosticar falhas em motores a diesel.

Desde Dunia e Qin (1998), o PLS geralmente é usado junto com o PCA para o diagnóstico de falhas. Em Vitale, Noord e Ferrer (2014) é feita uma combinação de técnicas baseadas em *kernel*, PCA e PLS, para fornecer uma discriminação eficiente das falhas em processos industriais.

A técnica da SVM também é um método baseado em análise estatística, capaz de obter alta eficiência com uma baixa quantidade de amostras. Namdari, Jazayeri-Rad e Hashemi (2014) usam um algoritmo genético para otimizar os parâmetros de uma SVM, e assim obter um resultado superior ao uso de redes neurais artificiais (RNA) na mesma aplicação.

Devido à capacidade de aproximação não linear e aprendizagem adaptativa, a RNA é uma das mais bem estabelecidas ferramentas de diagnóstico de falhas baseado em dados não estatísticos (GAO; CECATI; DING, 2015b). Diversas aplicações de RNA, para o diagnóstico de falhas podem ser encontradas, por exemplo, Elnokity et al. (2012) em processos nucleares e Valtierra-Rodriguez et al. (2014) detectando e classificando distúrbios de qualidade de energia. Os métodos da SVM e RNA juntamente com outras técnicas serão aprofundados na seção 3.4.

A lógica difusa ou *Fuzzy Logic* (FL) é uma abordagem de particionamento do espaço das variáveis em conjuntos nebulosos e uso de regras difusas para o raciocínio, a fim de criar um raciocínio aproximadamente humano. Desta forma, a FL é capaz de incorporar incertezas e possibilidades, que são muito úteis na tomada de decisão, e também para um processo de diagnóstico de falhas (GAO; CECATI; DING, 2015b; DAI; GAO, 2013). Nan, Khan e Iqbal (2008) apresentam uma integração entre um sistema especialista, e FL para a implementação de um algoritmo de diagnóstico de falhas em tempo real para aplicações industriais.

Algumas técnicas se tornaram mais populares nos últimos anos, o que é atribuído a um aumento no poder de computação e *big data*. O Deep Learning (DL) é uma destas técnicas, sendo capaz de extrair automaticamente características sem a necessidade de um especialista do sistema (KHAN; YAIRI, 2018). Existem várias arquiteturas de DL melhor detalhadas na subseção 3.4.5.

Na Figura 6, podem ser vistos fluxogramas comparando os métodos que fazem a extração automática de características com o DL e métodos convencionais, realizando a extração de características de forma manual.

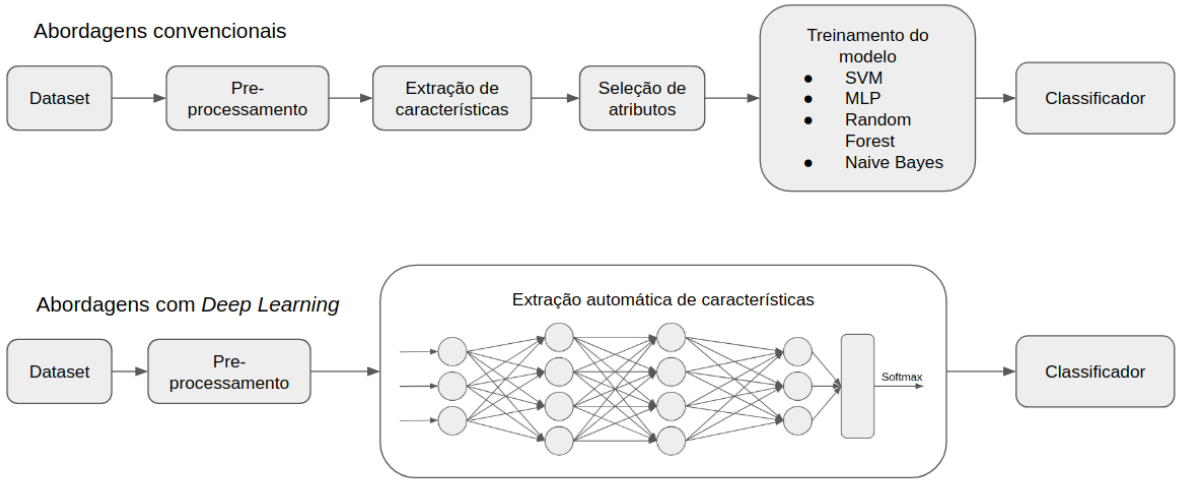


Figura 6 – Fluxograma de abordagens tradicionais e abordagens com DL.

Ainda na Figura 6, observa-se as etapas de extração de características e seleção de atributos presentes nas abordagens convencionais. Estas etapas são fundamentais para a redução de dimensionalidade, e visam garantir que o modelo de AM irá representar bem o sistema. O pre-processamento na figura é referente ao processo de limpeza, normalização e padronização dos dados.

2.5 Métodos híbridos

Cada uma das demais abordagens têm suas vantagens e desvantagens, assim, métodos híbridos integram dois ou mais métodos de diagnóstico de falhas. Os métodos baseados em modelos necessitam de poucos dados, no entanto precisam de um modelo matemático que represente o sistema. Os modelos baseados em sinais e conhecimento não têm a necessidade de um modelo matemático, o que facilita muito para sistemas onde modelos explícitos são muito custosos ou indisponíveis, mas geralmente é necessário uma grande quantidade de dados e geralmente estão atrelados a um alto custo computacional (GAO; CECATI; DING, 2015b).

A maioria das abordagens híbridas fazem uma mistura dos métodos baseados em sinais e conhecimento. Um exemplo disso é o estudo apresentado por Seshadrinath, Singh e Panigrahi (2014), o qual faz uso da transformada de *Wavelet* para extrair características do sinal de máquinas de indução, depois realiza um PCA, a fim de sumarizar os dados, e por fim, utiliza uma RNA para o diagnóstico.

2.6 Considerações finais

Segundo Witten et al. (2016) e Bishop (2006), nunca se gerou tanta informação como nos últimos anos e estima-se que a quantidade de dados armazenados dobra a cada vinte meses. Assim, há uma maior possibilidade de que sejam encontrados padrões e tendências, e com isso, pode-se transformar estes dados em informações úteis para as mais diversas aplicações.

Engenheiros, economistas, meteorologistas, estatísticos e outros, encontram vários padrões em suas pesquisas. Estes padrões nem sempre são intuitivos e simples de serem encontrados. Portanto, é necessário uma abordagem eficaz para alcançar este objetivo. Dessa forma, o aprendizado de máquina tem se demonstrado eficiente (DUDA; HART; STORK, 2012; WITTEN et al., 2016).

Liu et al. (2013) relatam que a essência da detecção e classificação de falhas é o reconhecimento de padrões e as condições de operação do sistema. Assim, no Capítulo 3, os principais conceitos de aprendizado de máquina serão abordados para serem aplicados ao diagnóstico de falhas.

3 Aprendizado de Máquina

3.1 Considerações iniciais

É perceptível que os seres humanos têm uma capacidade natural de aprendizado. Seres humanos nascem com esta habilidade, que pode ser desenvolvida, acumulando experiências durante a vida. O Aprendizado de Máquina (AM) é uma área de pesquisa da ciência da computação que procura desenvolver técnicas e métodos capazes de aprender com experiências (BISHOP, 2006). Mitchell (1997) define aprendizado de máquina como um processo de indução de uma hipótese a partir de experiências passadas.

Através de uma representação dos mais diversos tipos de dados em um ambiente computacional, as técnicas de AM podem gerar modelos aptos a organizar conhecimentos intrínsecos no conjunto de dados, ou até mesmo imitar o comportamento de um especialista no conjunto de dados a ser estudado. Desta forma, o objetivo é desenvolver métodos para que um computador possa aprender a partir de um conjunto de dados (CUPERTINO, 2014; DUDA; HART; STORK, 2012).

Os modelos criados pelas técnicas de AM podem realizar um diagnóstico de falhas em sistemas, uma vez que as situações de mau funcionamento do sistema são ditas como padrões. Assim, Duda, Hart e Stork (2012) definem o reconhecimento de padrões como o ato de utilizar um conjunto de dados para decidir uma ação baseada no padrão da categoria encontrada.

O reconhecimento de padrões abrange desde uma simples escolha entre dois objetos, até uma complexa realização de aprendizado, como a detecção de injeção de dados falsos em *smart grids* (HE; MENDIS; WEI, 2017). Além disso, observa-se que este conceito está presente no cotidiano das pessoas nas mais diversas situações (CAMPOS, 2001; JAIN; DUIN; MAO, 2000; HASTIE; TIBSHIRANI; FRIEDMAN, 2009; MORAIS, 2010).

3.2 Abordagens de aprendizado

Inicialmente, havia dois paradigmas: aprendizado supervisionado e aprendizado não supervisionado (MITCHELL, 1997). No aprendizado supervisionado, o objetivo é classificar dados usando outros dados já classificados, isto é, o processo de aprendizado envolve aprender a partir de um conjunto de treinamento, gerando um modelo, e aplicar este modelo em um conjunto de teste. Por outro lado, o aprendizado não supervisionado tem o objetivo de formar aglomerados de dados seguindo um critério de similaridade, sendo que esse critério é guiado unicamente pelos dados, já que não há nenhum conhecimento

prévio sobre as classes dos dados (JAMES et al., 2013).

Mais recentemente, um outro paradigma de AM, chamado aprendizado semi supervisionado, foi desenvolvido (CHAPELLE; SCHOLKOPF; ZIEN, 2006). Para Zhu e Goldberg (2009), as técnicas que fazem parte desse paradigma tentam superar a dificuldade de quando um método supervisionado necessita de classificações que são difíceis de serem obtidas. A ideia principal do aprendizado semi supervisionado é fazer a classificação, usando alguns dados previamente classificados e uma grande quantidade de dados não classificados. Essa abordagem pode resultar em uma precisão maior, além de necessitar de um esforço humano menor, e explorar um grande conjunto de dados não classificados.

3.3 Procedimento de classificação

A aplicação de um método de AM faz parte de um procedimento de classificação ou processo de mineração de dados (AZEVEDO; SANTOS, 2008). A importância de seguir um modelo para o procedimento de classificação, se dá devido a necessidade de um planejamento, *design* e execução do problema a ser abordado, que geralmente não é de natureza simples (KUMAR et al., 2017).

Existem vários modelos de procedimento que podem ser utilizados para fazer a classificação como, por exemplo o CRISP-DM, cujos passos envolvem o entendimento dos dados, preparação dos mesmos para posteriormente modelar e avaliar o modelo (CAMILO; SILVA, 2009). Há também o procedimento apresentado por Kumar et al. (2017), o qual se demonstrou robusto. Ele é detalhado a seguir e pode ser visto na Figura 7:

- **Problema**: Nesta etapa é necessário entender qual é o problema a ser resolvido. É nesta etapa que se define as classes que serão empregadas no processo de classificação;
- **Identificar dados necessários**: Uma análise criteriosa do conjunto de dados deve ser feita, a fim de detectar correlações entre os atributos, e entender de forma adequada, como os dados se comportam em cada situação existente. Assim, conhecendo os dados, é possível identificar quais dados são necessários e quais são obsoletos;
- **Limpar e rotular dados**: Após conhecer os dados, pode-se descartar aqueles considerados desnecessários, como registros com muitos atributos nulos ou erros grosseiros. Também aqui, rótulos podem ser atribuídos, caso necessário;
- **Pré-processamento de dados**: Podem existir atributos que necessitam ser normalizados, ou dados em forma de texto que devem ser convertidos para valores numéricos ou discretizados. Assim, nesta etapa, os dados podem sofrer algumas alterações para serem melhor utilizados pelo método de AM numa fase posterior;

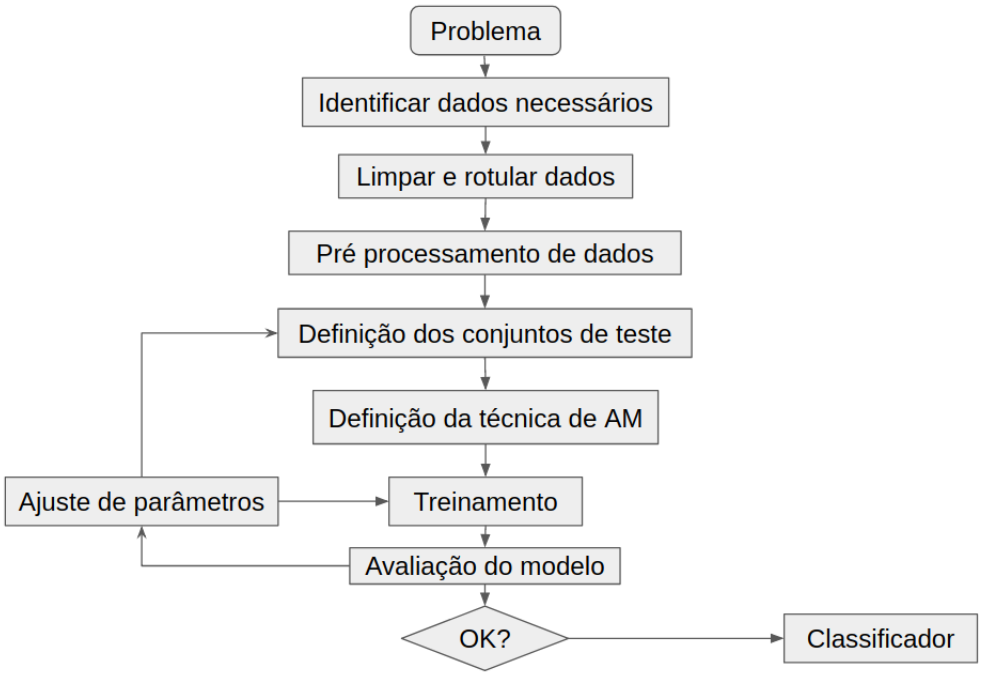


Figura 7 – Procedimento de classificação, adaptado de (KUMAR et al., 2017).

- **Definição dos conjuntos de teste**: A entrada de dados pode ser realizada de duas maneiras: validação cruzada e divisão do conjunto de dados em conjunto de teste e treinamento. Estas formas estão melhor explanadas na seção 3.5;
- **Definição da técnica de AM**: Como existem diversos métodos de classificação, um deve ser escolhido. Para tanto, alguns fatores, como tamanho do conjunto de dados e o tipo de cada atributo, são levados em consideração. As técnicas mais comuns são abordadas na seção 3.4;
- **Treinamento**: Na fase de treinamento, o algoritmo de AM escolhido é executado, treinando o modelo para classificar um conjunto de teste posteriormente;
- **Avaliação do modelo**: O modelo pode ser testado de acordo com a forma que o conjunto de testes foi definido. A eficiência do modelo treinado é calculada e caso ela seja aceitável, obtém-se um classificador pronto. Caso não seja aceitável é necessário fazer um ajuste nos parâmetros do modelo;
- **Ajuste de parâmetros**: Cada técnica de AM possui diversos parâmetros. Caso os parâmetros atuais não gerem um resultado satisfatório, uma alteração nos parâmetros pode apresentar um melhor resultado.

Para Olson e Delen (2008) é fundamental que o problema seja descrito de forma clara e formal, que os dados relevantes sejam identificados para a resolução do problema, e

também que as variáveis relevantes não sejam interdependentes. Assim, só após o cumprimento destes três requisitos o método de AM deve ser aplicado. Na seção 3.4, as principais técnicas de AM serão apresentadas.

3.4 Técnicas de Aprendizado de máquina

Existem na literatura várias técnicas de AM. O foco deste trabalho está nas técnicas de aprendizado supervisionado, devido ao formato do conjunto de dados a ser estudado e os objetivos desta pesquisa. Desta forma, alguns dos principais algoritmos de aprendizado supervisionado são abordados nesta seção.

3.4.1 Árvores de Decisão

Um dos algoritmos mais simples de aprendizado em máquina é a árvore de decisão. Essa técnica tem sido utilizada desde a década de 1960, quando Hunt, Marin e Stone (1966) utilizaram árvores de decisão para modelar o aprendizado de conceitos humanos. No entanto, foi na década de 1980, graças ao trabalho de Breiman et al. (1984), que o método se popularizou.

Segundo Perez (2012), uma árvore de decisão é uma estrutura de dados definida de forma recursiva, como:

- um nó folha que representa uma classe;
- um nó de decisão, que contém um teste sobre algum atributo. Para cada resultado do teste, existe uma aresta para uma subárvore.

O algoritmo básico de construção de uma árvore de decisão é considerado simples, conduzido de forma gulosa (*greedy*), ou seja, o algoritmo não considera escolhas anteriores (QUINLAN, 1986).

O funcionamento do algoritmo básico para a construção de uma árvore de decisão, consiste em: por meio de um conjunto de treinamento, um atributo é escolhido para particionar os exemplos em subconjuntos, esta escolha é feita minimizando a entropia dos subconjuntos criados, de acordo com valores desse atributo selecionado. Para cada subconjunto, um outro atributo deve ser escolhido para particionar novamente cada um deles. Assim, cada escolha de atributo representa um teste realizado em um nó da árvore, de forma que, cada nó interno execute um teste em apenas um atributo e tenha dois ou mais ramos, cada um representando um possível resultado do teste. Este processo segue enquanto um dado subconjunto contenha uma mistura de exemplos, com relação aos rótulos da classe. Quando é obtido um subconjunto uniforme, isto é, quando todos

exemplos no subconjunto pertencem a mesma classe, é criado um nó folha, sendo rotulado com o mesmo nome da respectiva classe. Para rotular um novo exemplo deve-se voltar ao nó raiz, realizar os testes utilizando os atributos do exemplo até chegar a um nó folha, assim o novo exemplo é classificado como o rótulo do nó folha (PEREZ, 2012; ROKACH, 2015). Oshiro (2013) alerta que caso uma árvore de decisão cresça demais, ela se tornará muito específica para a base de dados causando assim um problema de *overfitting* (sobre-ajuste).

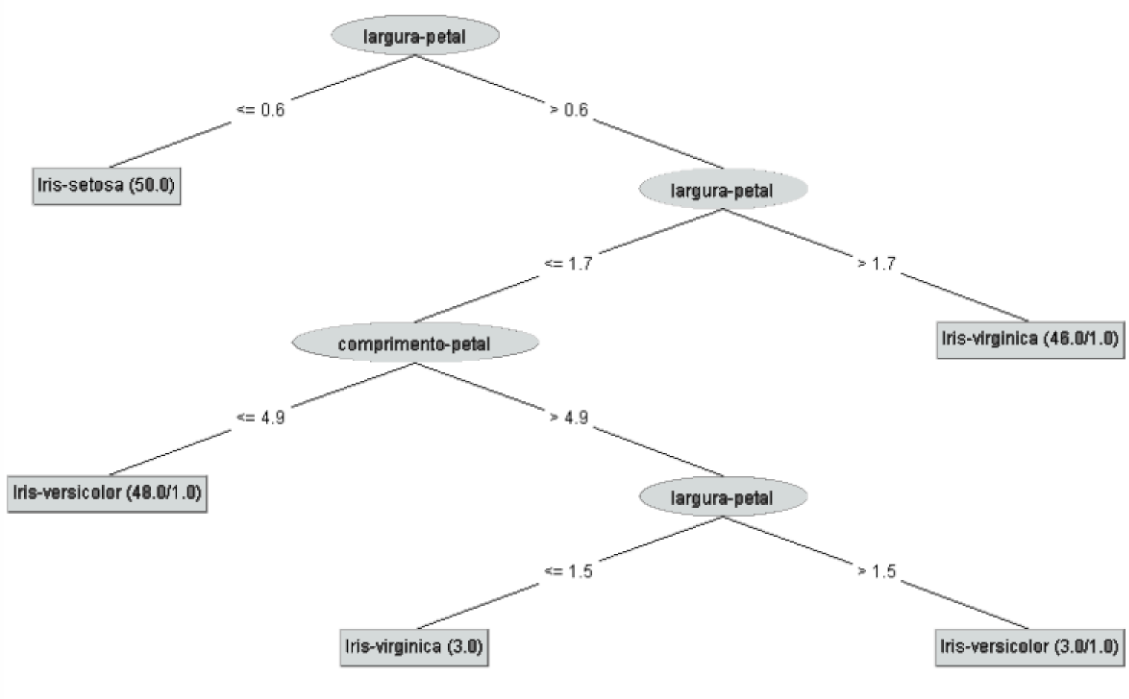


Figura 8 – Exemplo de árvore de decisão.

Para exemplificar, na Figura 8, pode-se observar uma árvore de decisão feita a partir do *software* Weka[1] (HALL et al., 2009), utilizando o banco de dados Iris, onde é possível classificar uma nova flor iris em uma das três classes do conjunto: Iris setosa, Iris virgínica e Iris versicolor.

Começando pela raiz da árvore, verifica-se que a largura da pétala da iris é menor ou igual a 0,6 (largura-petal $\leq$ 0,6). Caso este teste seja positivo, encontra-se um nó folha com o rótulo Iris-setosa, então a classificação da nova iris é encontrada. Caso o teste da raiz não retorne verdadeiro, um outro nó é encontrado e um novo teste é realizado. Neste teste, a largura da pétala é avaliada novamente agora verificando se ela é maior que 1,7 (largura-petal $>$ 1,7). Se este segundo teste for positivo, um outro nó folha com o rótulo

[1] Weka é uma coleção de algoritmos de AM escritos em Java para tarefas de mineração de dados. Ele contém ferramentas para preparação de dados, classificação, regressão, agrupamento, mineração de regras de associação e visualização.

iris-virgínica é encontrado. Caso o teste seja negativo, deve-se continuar percorrendo a árvore até encontrar um nó folha.

3.4.2 *Random Forest*

O método da *Random Forest* é uma variação do método da árvore de decisão apresentada em Breiman (2001). O método é um classificador composto por k árvores aleatórias, independentes e identicamente distribuídas, e cada árvore apresenta um resultado para a entrada desejada. A palavra aleatória significa que cada árvore tem uma chance igual de ser gerada. Estas florestas são capazes de gerar modelos bastante precisos (DUBATH et al., 2011; ZHAO; ZHANG, 2008; STIBOREK; PEVNỲ; REHÁK, 2017).

Cada árvore aleatória é feita a partir de um conjunto de árvores possíveis, fazendo o uso de a atributos aleatórios em cada nó, sendo $a \leq m$, com m sendo o número total de atributos. Para a classificação, cada árvore vota em uma classe para uma determinada entrada, assim, a entrada será classificada com a classe mais votada.

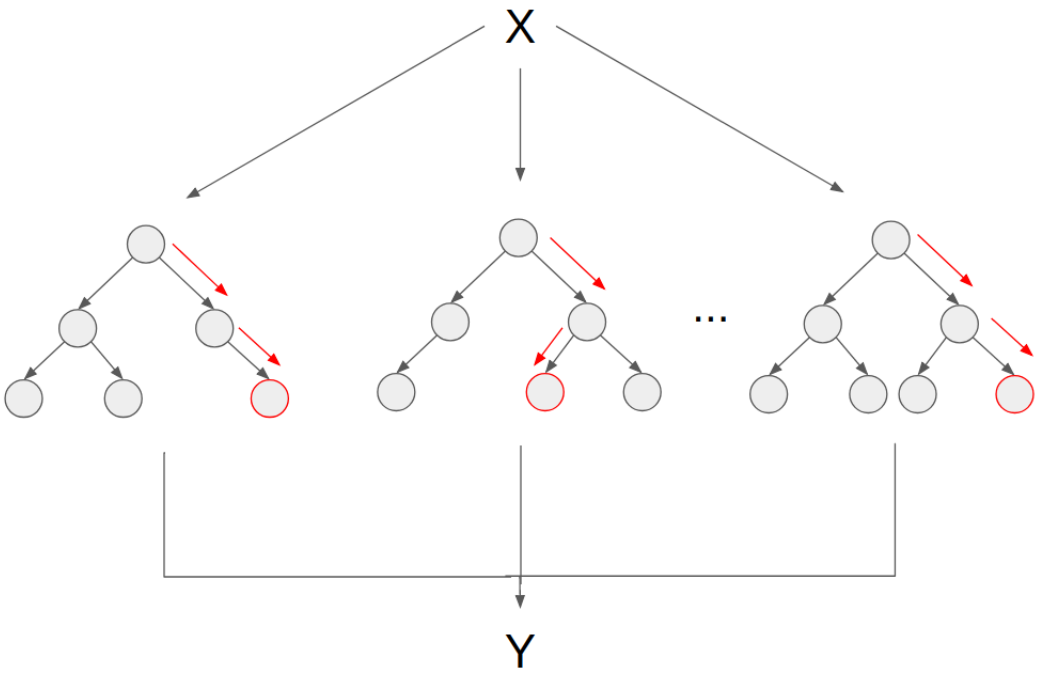


Figura 9 – Funcionamento da *Random Forest*.

Na Figura 9, o funcionamento da *Random Forest* pode ser visualizado. Pode-se ver que o mesmo registro é dado como entrada em diversas árvores de decisão aleatórias e cada uma delas pode apresentar uma classificação diferente. Ao final, a votação é realizada para classificar o registro de entrada.

A eficiência da *Random Forest* depende da força individual de cada árvore e da correlação entre quaisquer duas árvores (BREIMAN, 2001; MA; GUO; CUKIC, 2006):

- **Força individual de cada árvore**: Cada árvore tem uma chance de acerto, assim quando uma árvore tem uma taxa de acerto alta, esta tem uma força maior. Desta forma, quanto maior a força individual das árvores, maior a eficiência do classificador;

- **Correlação entre as árvores**: Como as árvores são geradas de forma aleatória, a correlação entre elas tende a não ser alta. A eficiência do classificador tende a aumentar, quando a correlação entre as árvores é menor.

3.4.3 Máquinas de Vetor de Suporte

Uma máquina de vetor de suporte ou *Support Vector Machine* (SVM) é uma técnica de aprendizado baseada no uso de *kernels* e regras não lineares (VAPNIK; VAPNIK, 1998). A SVM é uma técnica de classificação binária, realizando uma separação automática entre duas classes de características distintas (ALBUQUERQUE, 2012).

A SVM consiste em um método de classificação supervisionado, com objetivo de fazer uma separação ótima de classes (VAPNIK, 2013). No entanto, pode existir mais de uma função que separa as classes. Desta forma, deve-se avaliar qual é a função que melhor faz a distinção entre as categorias de dados. Para isso, segundo Vapnik (2013), a função ideal é aquela que apresenta a maior margem entre as classes analisadas. Para auxiliar o entendimento do conceito de margem, um novo conceito foi criado, no caso, o vetor de suporte que dá nome a técnica.

Os vetores de suporte (*Support Vectors*) são pontos de ambas classes que estão mais próximos do separador de classes e o número de vetores de suporte é sempre menor que a quantidade total de amostras (VAPNIK, 2013). De acordo com Lorena e Carvalho (2007), o separador ótimo é aquele que apresenta a maior margem possível entre a função de separação e os vetores de suporte.

A função de separação, também pode ser chamada de hiperplano de separação (BURGES, 1998). Um exemplo deste hiperplano de separação pode ser visto na Figura 10, com um conjunto de dados em R^3. Desta maneira, o objetivo é encontrar um hiperplano de separação ótimo, para tal, deve-se resolver o problema de otimização, representado através das Equações 3.1 e 3.2, onde $\mathbf{x}_i$ é um conjunto de treinamento, y_i seus rótulos, $\mathbf{x}_j$ é o conjunto de teste, y_j seus respectivos rótulos, e α_i e α_j são os multiplicadores de *Lagrange* dos conjuntos de treinamento e teste (LORENA; CARVALHO, 2007).

$$\text{Maximizar} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x_i} \cdot \mathbf{x_j}) \tag{3.1}$$

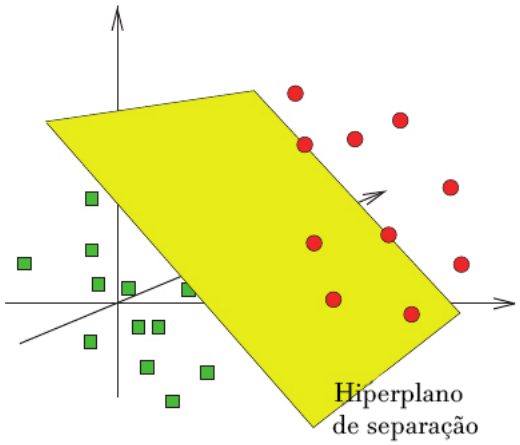


Figura 10 – Hiperplano de separação (HUSON, 2007).

$$\text{Com as restrições:} \quad \begin{cases} \alpha_i \geq 0, \forall i = 1, ..., n \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases} \tag{3.2}$$

Como os dados coletados na vida real geralmente não são linearmente separáveis, Hofmann, Schölkopf e Smola (2008) tentaram resolver o problema da não-linearidade dos dados através de uma re-projeção dos dados amostrais em um espaço dimensional, maior por meio de uma função de Kernel (Φ), e assim, neste novo espaço amostral, aplicar o algoritmo de classificação proposto por Vapnik e Vapnik (1998). Um exemplo de aplicação de uma função de Kernel pode ser verificado na Figura 11, na qual um conjunto de dados em R^2 é transformado em um novo conjunto em R^3.

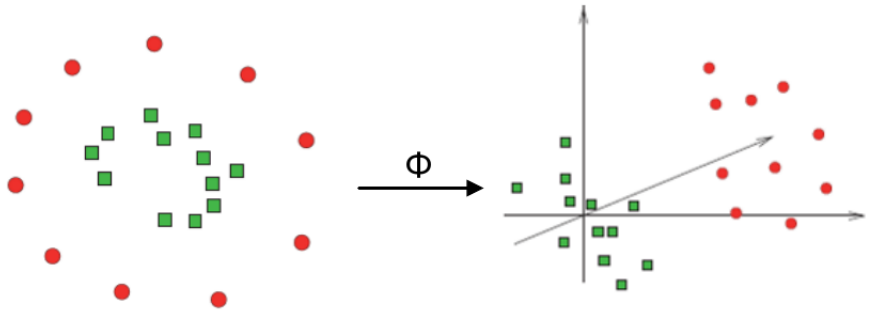


Figura 11 – Aplicação de uma função de Kernel (HUSON, 2007).

Um Kernel equivale a um produto escalar encontrado em um espaço de dimensão superior ao de origem dos atributos. Assim, a aplicação de um Kernel otimiza a atuação do separador de classes (HOFMANN; SCHÖLKOPF; SMOLA, 2008). É comum que pes-

quisadores proponham novas funções de Kernel, no entanto, as quatro funções básicas são listadas por Hsu et al. (2003), onde γ, r e d são parâmetros da função de Kernel:

- **Linear**: $K(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i^T x_j}$;
- **Polinomial**: $K(\mathbf{x_i}, \mathbf{x_j}) = (\gamma \mathbf{x_i^T x_j} + r)^d, \gamma > 0$;
- **Função de base radial ou *radial basis function* (RBF)**: $K(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \parallel \mathbf{x_i} - \mathbf{x_j} \parallel^2), \gamma > 0$;
- **Sigmoid**: $K(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\gamma \mathbf{x_i^T x_j} + r)$.

Além destes parâmetros, um outro parâmetro C é definido como parâmetro de penalidade do algoritmo, o qual deve ser maior que zero. A demonstração matemática da existência deste parâmetro no problema de otimização apresentado nas equações 3.1 e 3.2, é mostrado em Burges (1998) e Lorena e Carvalho (2007). Este parâmetro diz o quanto o modelo será tolerante a erros. Note que, dependendo do valor de C, o modelo pode ter problemas de *overfitting* (sobre-ajuste) ou *underfitting* (sub-ajuste).

Para escolher todos estes parâmetros, geralmente é realizada uma busca em grade (*grid search*), isto é, são feitos vários treinamentos com parâmetros diferentes escolhidos previamente, em seguida os resultados são comparados e define-se quais foram os parâmetros que obtiveram um resultado mais próximo ao desejado (HSU et al., 2003). Este método também é conhecido como método exaustivo. Também é possível otimizar estes parâmetros com um método evolutivo, empregando um algoritmo genético ou um enxame de partículas (POURBASHEER et al., 2009; LIN et al., 2008).

Como já mencionado, o SVM é um classificador binário. No entanto, diversas aplicações têm a necessidade de classificadores multi-classe. Para resolver este desafio, existem duas abordagens: os métodos "um contra um" e "um contra todos".

No método "um contra um" são feitos diversos classificadores formados por todos os pares de classes, cada um com suas regras. Para cada instância a ser classificada, cada par de classes vota acerca de qual classe a instância avaliada pertence, assim este objeto será classificado com a classe mais votada. Já no método "um contra todos", é feito um classificador para cada classe, e valores de probabilidade, são calculados. Assim, a classe que tiver o maior valor de probabilidade determinará o resultado da classificação. O método "um contra todos" é computacionalmente mais rápido, no entanto geralmente, apresenta eficiência inferior ao método "um contra um" (HSU; LIN, 2002).

3.4.4 *Naive Bayes*

Este método é fundamentado na probabilidade condicional, especialmente no teorema de Bayes, enunciado posteriormente. Assim, este método é baseado na probabilidade

de um evento ocorrer, dada a probabilidade de outros determinados eventos já ocorrerem (GRUS, 2018).

O algoritmo *Naive Bayes* faz um modelo probabilístico, baseado em conhecimento prévio do problema, combinando com exemplos de treinamento, para determinar qual é a classificação provável de cada registro (MITCHELL, 1997).

Considerando a probabilidade de um evento A acontecer, dada a probabilidade de um outro evento independente B acontecer, a probabilidade condicional é dada por $P(A|B)$. O teorema de Bayes é enunciado pela Equação 3.3 (BUSSAB, 2010). Nela, pode-se calcular a probabilidade de um evento A_i acontecer, dada a probabilidade de um outro evento B independente acontecer previamente.

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)} \tag{3.3}$$

Nota-se que para usar a equação de Bayes, os eventos devem ser independentes. Assim, Gallagher, Madden e D'Arcy (2015) dizem que o método *Naive Bayes* tem esse nome, justamente porque ele assume que todos eventos são independentes entre si.

Este método cria uma rede bayesiana, isto é, um grafo acíclico direcionado mostrando as relações de causalidade entre as variáveis (CHARNIAK, 1991). Um exemplo de uma rede bayesiana pode ser visto na Figura 12, na qual vários nós atributos (X_i) estão ligados à classe Y, mas nenhum atributo está ligado a outro atributo, o que mostra a independência entre os atributos.

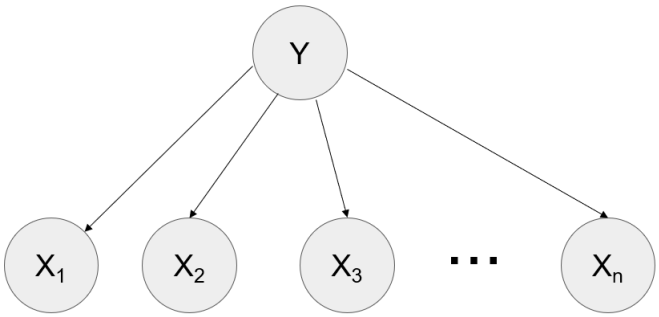


Figura 12 – Exemplo de uma rede bayesiana usada para o método *Naive Bayes*.

O *Naive Bayes* faz a classificação de um registro associando este à classe mais provável (Y_{map}). Para calcular esta probabilidade condicional é usada a Equação 3.4, que determina a probabilidade de cada classe Y_j acontecer dada a probabilidade dos atributos do registro (X_i) acontecerem (GONÇALVES, 2002).

$$Y_{map} = max_{y_j \in \mathbf{Y}} P(Y_j|X_1, X_2, ..., X_n) \tag{3.4}$$

Nota-se que, a Equação 3.4 pode ser reescrita usando a Equação 3.3:

$$Y_{map} = max_{y_j \in \mathbf{Y}} \frac{P(X_1, X_2, ..., X_n|Y_j)P(Y_i)}{P(X_1, X_2, ..., X_n)} \tag{3.5}$$

Além disso, como os atributos são considerados independentes, a Equação 3.5 pode ser reescrita com um produtório:

$$Y_{map} = max_{y_j \in \mathbf{Y}} P(Y_j) \prod_{i=0}^{n} P(X_i|Y_j) \tag{3.6}$$

Assim, para calcular a probabilidade de uma classe P(Y_j), deve-se determinar a frequência de cada Y_i no conjunto de treinamento. No entanto, quando o conjunto de dados é grande, pode ser muito custoso fazer esta análise, já que o número de termos analisados é igual ao número de atributos multiplicado pelo número de classes. Pode ser necessário analisar várias vezes cada exemplo para obter uma estimativa aceitável.

3.4.5 Redes Neurais Artificiais

Uma das técnicas mais comuns do AM são as Redes Neurais Artificiais (RNA), que são compostas de neurônios artificiais representando o funcionamento biológico (MCCULLOCH; PITTS, 1943). O funcionamento de um neurônio artificial consiste em uma combinação linear dos sinais de entrada, e a aplicação de uma função de ativação para gerar a saída do neurônio. Este comportamento pode ser visualizado na Figura 13.

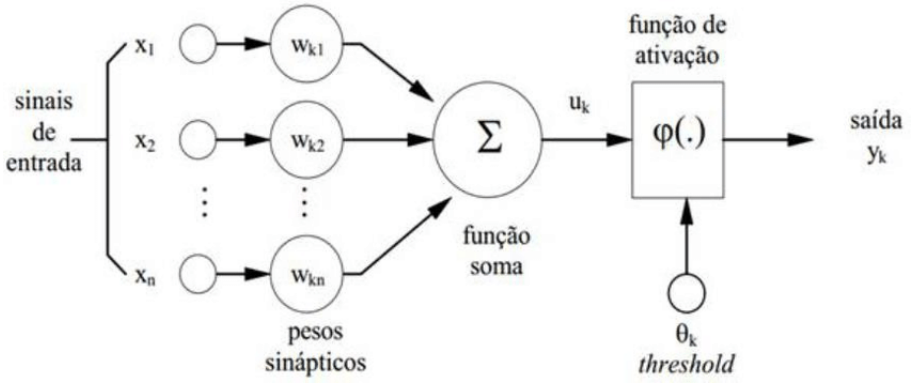


Figura 13 – Funcionamento de um neurônio artificial (HAYKIN, 2007).

Haykin (2007) descreve o modelo de forma que cada entrada (x_i) é associada a um peso (w_i). Na Equação 3.7, o resultado da combinação linear de todas entradas é dado como entrada em uma função de ativação (φ), e somado com um valor de viés, que pode ser visto como um peso não vinculado aos dados de entrada.

$$y = \varphi(\sum_{j=1}^{n} x_j w_j + b) \tag{3.7}$$

Apesar da simplicidade do modelo neural, ele é capaz de produzir resultados bons, baseando-se na possibilidade de criar várias camadas com até mesmo milhares de neurônios. Os neurônios da primeira camada recebem os sinais de entrada, processam esta informação e a propagam para a próxima camada. Esta operação se repete até o último neurônio da última camada.

O processo de aprendizado em uma RNA consiste na troca dos pesos, baseando-se nos dados do conjunto de treinamento até encontrar o melhor valor para cada peso a fim de minimizar a diferença entre as saídas da rede e os valores objetivo (HAYKIN, 2007). Depois deste modelo ser proposto, várias técnicas de treinamento foram feitas. Uma das mais conhecidas é o algoritmo *Backpropagation*, proposto por Rumelhart, Hinton e Williams (1988). Este método calcula o gradiente da função a ser minimizada (geralmente a diferença entre as saídas da rede e os valores objetivo) a respeito de todos os pesos na rede, e então atualiza cada peso a fim de minimizar esta função. No momento da atualização de cada peso, o gradiente é multiplicado por um valor α, denominado taxa de aprendizado (*learning rate*). Esta multiplicação tem como objetivo controlar a velocidade de aprendizado das RNAs.

Uma das arquiteturas de RNA mais simples é o Perceptron de múltiplas camadas, ou *Multi Layer Perceptron* (MLP) (CHURCHLAND; SEJNOWSKI; POGGIO, 2016). Geralmente uma MLP possui três camadas: uma recebendo os dados de entrada, uma camada oculta e uma camada de saída, sendo que todas camadas são densamente conectadas, ou seja, todos neurônios da camada anterior têm conexão com a sua camada posterior. Para ajustar um modelo do tipo MLP, devem ser considerados diversos parâmetros como o número de neurônios de cada camada, o número de camadas da rede, a taxa de aprendizado (α), além da função de ativação associada a cada neurônio.

Por algumas décadas, vários pesquisadores foram motivados a encontrar formas de treinar redes neurais profundas, com mais de duas camadas ocultas, isso possivelmente resultaria em modelos mais precisos (BENGIO; LECUN et al., 2007; UTGOFF; STRACUZZI, 2002). Hinton (2007) mostrou como uma rede neural de várias camadas poderia ser pré-treinada, uma camada de cada vez, e depois fazer ajustes com o método *backpropagation* supervisionado.

Segundo Goodfellow, Bengio e Courville (2016), *Deep Learning* (DL) ou aprendizado profundo, é um ramo do AM que faz o uso de redes neurais com mais de duas camadas ocultas, como pode ser visto na Figura 14. Este conjunto de técnicas promete substituir os processos feitos de forma manual com algoritmos eficientes, para o aprendizado semi supervisionado e não supervisionado.

Existem várias arquiteturas de DL, como redes neurais profundas, redes neurais convolucionais, redes neurais recorrentes e suas extensões como a rede LSTM (*Long-Short Term Memory*). Todas estas são utilizadas para diversas finalidades, incluindo visão com-

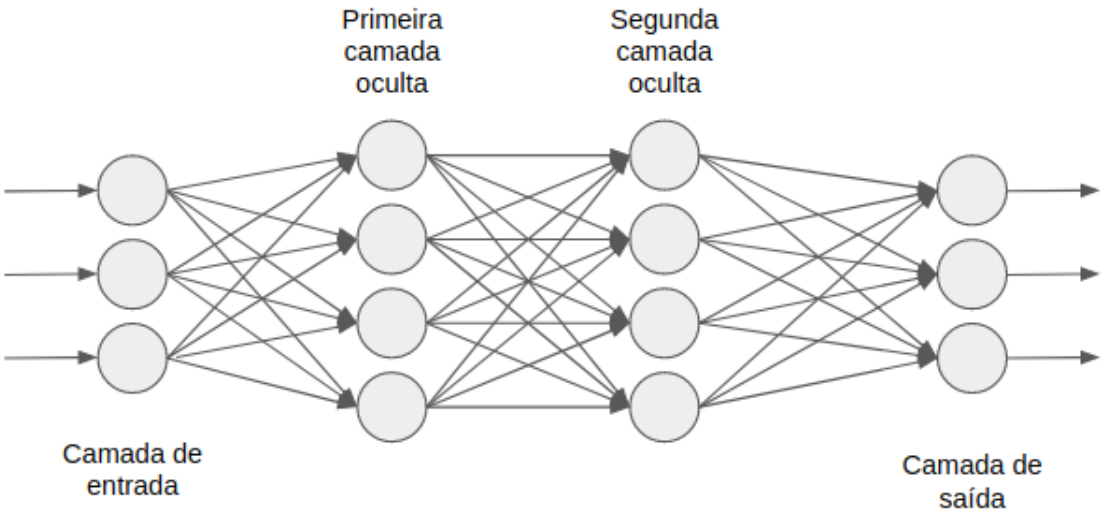


Figura 14 – Rede neural com duas camadas ocultas.

putacional, reconhecimento de fala, processamento de linguagem natural, reconhecimento de áudio e imagens, dentre outras (LECUN; BENGIO; HINTON, 2015; SCHMIDHUBER, 2015).

3.4.5.1 Redes Neurais Convolucionais

A primeira Rede Neural Convolucional ou *Convolutional Neural Network* (CNN) foi proposta por LeCun, Bengio et al. (1995). No entanto, apenas em 2012 estas redes ganharam destaque, isto devido ao trabalho de Krizhevsky, Sutskever e Hinton (2012) que revolucionou as técnicas de classificação de imagens.

Um exemplo clássico deste tipo de rede neural é a LeNet-5, proposta por LeCun et al. (1998). Esta rede possui camadas convolucionais, sub-amostragem ou *pooling* e camadas densamente conectadas. As camadas convolucionais juntamente com as de sub-amostragem são responsáveis pela extração de características, e as camadas densamente conectadas fazem um papel semelhante ao de uma rede MLP simples, como pode ser visto na Figura 15.

As camadas convolucionais extraem as características dos registros. Para tal, elas possuem uma quantidade n de filtros, que carregam as variáveis que serão treinadas. Estas camadas são chamadas convolucionais, já que são feitas convoluções entre a entrada e os filtros presentes nas camadas convolucionais.

As camadas de sub-amostragem podem remover atributos redundantes, além de fazer com que o modelo seja menos sensível a translação, rotação e dimensionamento. Para isto, é definida uma janela de sub-amostragem e uma operação a ser aplicada nesta janela. Esta operação geralmente é de valor máximo (*Max-pooling*), ou seja, escolhe o maior valor

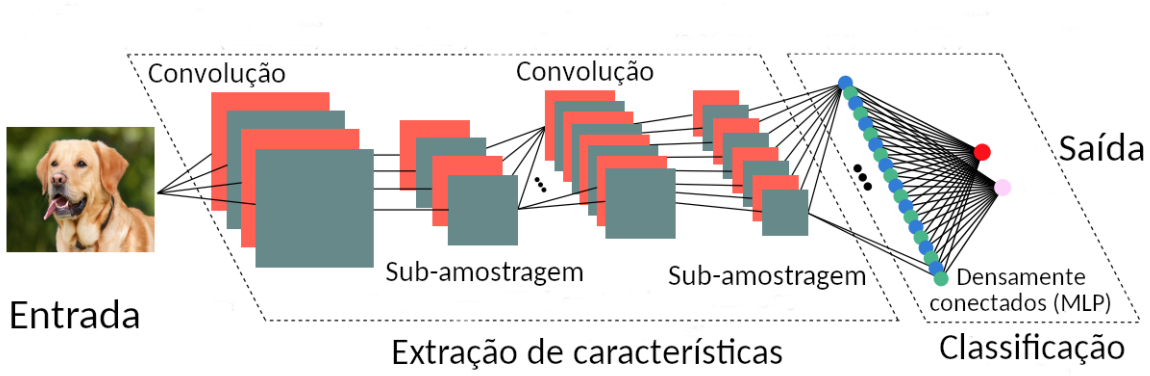


Figura 15 – Exemplo de CNN, adaptado de (MENDES, 2017).

presente na janela. No entanto, esta operação também pode ser a média dos valores na janela (*Avg-pooling*), como é ilustrado na Figura 16, onde observa-se uma entrada 4x4 e uma janela 2x2.

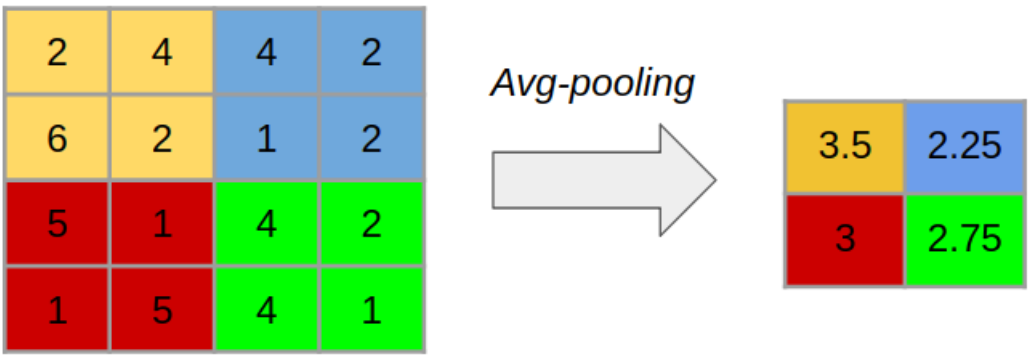


Figura 16 – Exemplo de funcionamento de uma camada de sub-amostragem.

Para ligar as camadas convolucionais e de sub-amostragem às camadas densamente conectadas, é necessário que haja uma redução de dimensionalidade. Desta forma, existe uma camada de achatamento ou *flatten*, que transforma os dados multidimensionais, da última camada de sub-amostragem em dados com apenas uma dimenção, podendo assim ser ligados as últimas camadas densamente conectadas, compostas de neurônios, como uma MLP simples (TAN et al., 2018).

Para evitar sobre-ajustes nos modelos neurais, uma das abordagens comuns é o uso de camadas *Dropout*. Estas camadas desativam alguns dos neurônios da camada anterior de forma aleatória, por exemplo. Caso haja uma camada densamente conectada com n neurônios, e logo após esta camada exista uma camada de *Dropout* de 50%, cada neurônio tem uma chance de 50% de ser inutilizado. Nota-se que os neurônios inutilizados podem mudar a cada iteração do treinamento.

Assim, como nas tradicionais MLPs, as CNNs também possuem funções de ativação ao final das camadas convolucionais, e densamente conectadas. A função de ativação usada na maioria dos trabalhos clássicos é a função *sigmoid* (MITCHELL, 1997), definida pela Equação 3.8:

$$f(x) = \frac{1}{1 + e^{x-1}} \tag{3.8}$$

No entanto, trabalhos mais recentes fazem o uso da função unidade linear retificada, ou *Rectified Linear Unit* (ReLU), definida pela Equação 3.9. Esta função evita números negativos, e assim é capaz de treinar redes neurais mais rapidamente sem perdas significativas (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Outra função de ativação importante é a função *softmax*, usada na última camada densamente conectada de redes neurais profundas. Esta última camada tem quantidade de neurônios igual a quantidade de classes do problema, e a função *softmax*. Cada neurônio da camada de *softmax* se associa a uma classe, e assim sua saída é referente a probabilidade da entrada ser da classe relativa ao neurônio.

$$f(x) = max(0, x) \tag{3.9}$$

Ainda existem diversos algoritmos para realizar a minimização da função de custo. O otimizador clássico é o gradiente descendente estocástico, ou *Stocastic Gradient Descent* (SGD) (GOODFELLOW; BENGIO; COURVILLE, 2016). No entanto, grande parte do trabalho moderno de otimização visa projetar regras de otimização, adaptadas para cada circunstância (ANDRYCHOWICZ et al., 2016). Neste contexto, melhorias no SGD como o momento de Polyak (1964) aplicado ao SGD, e momento de Nesterov (LAN, 2012) foram realizadas e também diversos outros otimizadores foram desenvolvidos a partir do SGD, Adagrad (DUCHI; HAZAN; SINGER, 2011), RMSprop (TIELEMAN; HINTON, 2017) e Adam (KINGMA; BA, 2014).

No treinamento destes modelos também é necessário definir uma quantidade de dados de entrada na rede neural. Este número é conhecido como *batch size*. Sendo assim, cada iteração na rede neural usa exatamente esta quantidade de dados. Se o *batch size* é grande, o treinamento da rede tende a ser mais rápido, no entanto, mais recursos computacionais são exigidos.

A técnica da CNN é poderosa e consegue ter um alto desempenho. No entanto, devido a quantidade de hiper-parâmetros, o processo de otimização de uma CNN é custoso e demanda tempo. Além disso, a rede trabalha como uma caixa preta, ou seja, não é possível extrair regras de classificação de uma CNN já treinada.

O processo de extração de características pela CNN, geralmente demonstra resultados superiores as técnicas tradicionais, como o SVM (WIATOWSKI; BÖLCSKEI,

2018). Para alguns casos, é possível visualizar os filtros presentes nas camadas convolucionais. Na Figura 17, pode-se observar os filtros de uma CNN para reconhecimento facial[2]. Os filtros da primeira camada representam detectores de borda e similares, enquanto os das camadas intermediárias representam olhos, narizes, bocas, orelhas; e finalmente, as últimas camadas convolucionais possuem filtros capazes de identificar rostos (SHULBY, 2018).

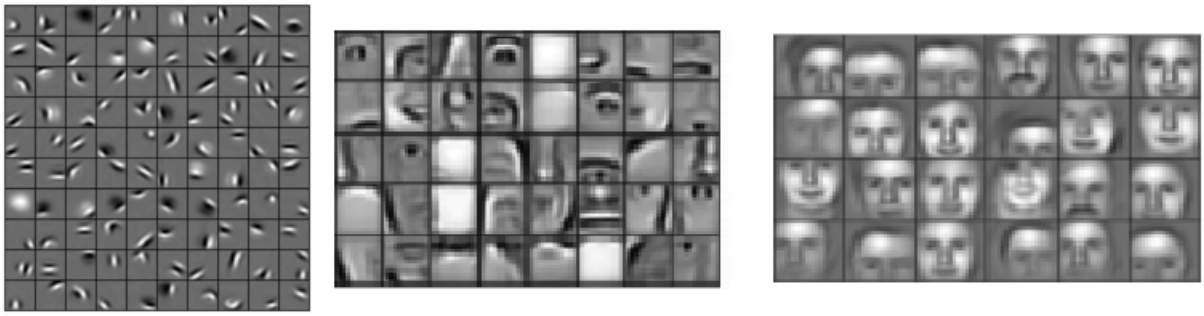


Figura 17 – Visualização das características no filtros de convolução.

3.5 Avaliação das técnicas

Para avaliar os métodos descritos na seção 3.4, alguns critérios, como acurácia, são utilizados para medir o desempenho destas técnicas. Para Kuncheva (2004), a avaliação de uma técnica de AM também deve envolver a precisão, ou seja, a habilidade de identificar *outliers* (valores discrepantes).

Para realizar o cálculo da acurácia, deve-se dividir o conjunto de dados em conjunto de treinamento e teste. Nota-se que, a medida da acurácia em cada um destes conjuntos é diferente, e que na maioria dos casos o resultado no conjunto de treinamento é superior. O conjunto de treinamento é usado para treinar o classificador, enquanto que o conjunto de teste é formado por dados nunca vistos anteriormente pelo classificador. Desta forma, o conjunto de teste é usado para avaliar o método (DUDA; HART; STORK, 2012).

Para dividir os dados nos conjuntos de treinamento e teste, são mais comuns o uso de duas abordagens (DUDA; HART; STORK, 2012; MICHIE et al., 1994):

- **Porcentagem do conjunto original**: divide-se o conjunto original de dados em dois. Um para o treinamento e outro para o teste. Na criação do conjunto de teste, deve-se excluir os dados da classe, para que o algoritmo calcule qual é a classe de cada instância;

[2] https://devblogs.nvidia.com/wp-content/uploads/2015/11/hierarchical_features.png

- **Validação cruzada**: consiste em dividir o conjunto original de dados em n subconjuntos, em seguida utilizar $n-1$ subconjuntos para gerar os classificadores, e o subconjunto restante é empregado para teste. Este processo é repetido para cada um dos n subconjuntos, assim todos os subconjuntos são usados para o treinamento e teste. Uma dificuldade da validação cruzada é que, como o processo se repete n vezes, o custo computacional pode se tornar alto.

Para avaliar os modelos, várias métricas podem ser utilizadas. Dentre estas, a acurácia é uma medida de desempenho muito comum e de fácil entendimento, no entanto avalia os métodos de forma geral, o que nem sempre é útil para melhorar o desempenho do mesmo. Ela é definida pela razão entre a quantidade de acertos do método, e a quantidade de dados avaliados. Sendo assim, o valor ideal de acurácia é 1, enquanto que o pior valor de acurácia é 0 e o valor esperado é $1/n$, sendo n o número de classes do problema (FACELI et al., 2011).

Outras medidas como a precisão e a sensibilidade são utilizadas para avaliar o modelo de forma menos geral. A precisão é definida pela Equação 3.10, onde VP é a quantidade de verdadeiros positivos, e FP é a quantidade de falsos positivos. No cenário do diagnóstico de falhas, esta medida leva ao questionamento de quantos dos registros classificados como defeituosos, quais realmente apresentam falhas.

$$prec(\hat{f}) = \frac{VP}{VP + FP} \tag{3.10}$$

A sensibilidade, também conhecida como *recall* ou taxa de verdadeiros positivos (TVP), é definida pela Equação 3.11, onde FN é a quantidade de falsos negativos. Também, no cenário do diagnóstico de falhas, esta medida conduz a seguinte reflexão: "quando um registro é da falha X, quantos são classificados como X?"

$$TVP(\hat{f}) = \frac{VP}{VP + FN} \tag{3.11}$$

Para avaliar um modelo de forma geral, usando a sensibilidade e a precisão, a medida-f é definida pela Equação 3.12 (FACELI et al., 2011). Nota-se que, esta medida é a média harmônica de peso 1 da precisão e da sensibilidade.

$$F_1(\hat{f}) = \frac{2 \times prec(\hat{f}) \times TVP(\hat{f})}{prec(\hat{f}) + TVP(\hat{f})} \tag{3.12}$$

Além destes critérios citados, uma forma comum e eficiente de avaliar estas técnicas é a matriz de confusão. Tan et al. (2018) descreve esta matriz de forma que cada elemento x_{ij}, corresponde ao número de elementos identificados como da j-ésima classe, classificados pelo modelo como pertencentes à i-ésima classe. Assim, o número de acertos do modelo

encontra-se na diagonal principal da matriz, e os demais elementos representam erros de classificação. Uma vantagem de analisar esta matriz é que, além de fornecer a exatidão do método numa visão geral, ela também permite examinar o desempenho do modelo para classes individuais.

Para que os modelos de AM tenham um bom aprendizado, e consequentemente apresentem boas métricas, é bom que o conjunto de dados esteja balanceado. Este conceito é melhor explicado na seção 3.6.

3.6 Dados desbalanceados

É comum que as bases de dados a serem trabalhadas estejam desbalanceadas, ou seja, que uma quantidade discrepante de dados de uma ou mais classes em relação com as demais classes seja utilizada. Por exemplo, em um problema com duas classes, se a classe positiva possui 80% dos dados, e a classe negativa possui 20%, esta base de dados pode ser considerada desbalanceada (RODRIGUES, 2015). Geralmente, uma base de dados é considerada desbalanceada quando uma das classes possui o dobro de elementos em relação a qualquer outra classe (PRATI; BATISTA; SILVA, 2015; FACELI et al., 2011).

Segundo Batista, Prati e Monard (2004), esta quantidade de dados desbalanceada pode fazer com que os algoritmos classificadores de AM criem uma tendência de classificação para a classe com maior quantidade de dados, chamada de classe majoritária. Para evitar este problema, geralmente o conjunto de dados é redefinido ou é feito o treinamento para apenas uma classe.

O tamanho do conjunto de treinamento pode ser redefinido mudando a quantidade de dados de cada classe. Para contornar o problema do desbalanceamento, é comum diminuir a quantidade de dados da classe majoritária ou aumentar a quantidade da classe minoritária. Caso necessário, é possível gerar dados artificiais para aumentar a quantidade de dados de uma classe minoritária.

Para realizar sobre-amostragens, uma das técnicas comuns é o SMOTE (Synthetic Minority Over-sampling Technique) (CHAWLA et al., 2002), e para a sub-amostragem, uma das técnicas comuns é o RUS (*Random Undersampling*) (SMITH et al., 2012), que seleciona de forma aleatória registros das classes majoritárias para serem descartados.

Ao aumentar a quantidade de dados, deve-se tomar cuidado para não gerar um problema de *overfitting*, que é um super ajuste do classificador ao conjunto de treinamento. Este problema geralmente acontece quando o processo a ser analisado é complexo, possuindo vários parâmetros e, assim, o classificador fica super ajustado até mesmo aos ruídos dos dados de treinamento. Alternativamente, quando se diminui a quantidade de dados da classe majoritária, pode acontecer o *underfitting*, que ocorre quando o classi-

ficador não consegue representar os dados do conjunto de treinamento (RODRIGUES, 2015).

3.7 Trabalhos correlatos

Os métodos de diagnóstico de falhas baseados em conhecimento fazem um amplo uso das técnicas de AM, abordadas neste Capítulo. Desta forma, nesta seção são apresentados trabalhos relacionados ao uso destas técnicas de AM, aplicadas ao diagnóstico de falhas.

Com o crescimento da quantidade de dados, muitos pesquisadores estão desenvolvendo aplicações para encontrar padrões nestes dados, inclusive identificar e classificar falhas. Dessa forma, com o avanço da computação, a abordagem de diagnóstico de falhas baseada em conhecimento quantitativo está cada vez mais comum.

Vários estudos disponíveis na literatura fazem o uso do aprendizado de máquina para a identificação e classificação de falhas (LIVANI; EVRENOSOĞLU, 2012; GOPAKUMAR; REDDY; MOHANTA, 2015; LIU et al., 2013; HESSINE; JOUINI; CHEBBI, 2014). Portanto, no estudo realizado neste projeto, o aprendizado de máquina foi utilizado para este fim.

Liu et al. (2013) propuseram um modelo para a classificação de múltiplas falhas em rolamentos. Na metodologia proposta, há uma fase de extração de características (domínios de tempo e frequência), onde o sinal de entrada é convertido em apenas 8 atributos. Em seguida, a SVM é usada para diagnosticar as falhas, para otimizar os parâmetros da SVM, um algoritmo de enxame de partículas é usado. Os resultados apresentados foram comparados com outros métodos tradicionais, e tiveram melhor generalização e robustez. No entanto, essa abordagem exigia a extração manual de recursos, e em comparação com trabalhos que extraem recursos automaticamente, eles apresentam resultados superiores, como ainda pode ser visto nesta seção.

Também para diagnosticar falhas em rolamentos, Gan, Wang et al. (2016) propuseram uma rede de diagnósticos hierárquicos, em inglês *Hierarchical Diagnosis Network* (HDN), que coleta redes de crenças profundas (*Deep Belief Networks*) por camadas para identificação hierárquica do sistema mecânico. *Wavelet Packet Transform* (WPT) foi usada para extrair recursos para as redes de crenças profundas. Os resultados foram comparados a um SVM e uma MLP, para confirmar a capacidade da HDN, mostrando que a HDN é confiável e pode superar os problemas causados por ruídos e outros distúrbios.

Lei, Liu e Jiang (2019) criaram um modelo de diagnóstico de falhas para detectar e identificar falhas em turbinas eólicas. Os autores utilizaram dados de quatro séries temporais adquiridas por quatro sensores, como entrada em uma rede LSTM, sem qualquer

extração manual de recursos, para realizar o diagnóstico de falhas. Comparações com outros métodos foram realizadas para avaliar o modelo, incluindo MLP, SVM, CNN e Rede Neural Recorrente (RNN), mostrando que o modelo LSTM foi superior em quase todas as métricas.

Para o diagnóstico inteligente de máquinas rotatórias, Jia et al. (2016) identificaram as deficiências de abordagens passadas, e também propuseram um modelo de classificador, baseado em *Deep Learning*. Os autores discutem as melhorias no uso de várias camadas ocultas, e seu grande potencial com uma quantidade massiva de dados. Para demonstrar a superioridade do modelo profundo, também foi feita uma comparação de eficiência com os métodos antigos.

Zhang et al. (2017) propuseram um modelo de rede neural profundo e raso, ou *Deep & Shallow Method* (DSM), combinado para aproveitar as vantagens das arquiteturas profundas e as rasas. Esta estrutura de modelo é uma rede neural linear simples, combinada com uma rede densa de alimentação profunda. Os resultados mostraram que o treinamento conjunto melhorou significativamente o desempenho de partes profundas e rasas. Além disso, para o cenário de caso de teste, o DSM teve um desempenho melhor do que CNN, LSTM, *Naive Bayes*, *Random Forest* e outros métodos.

Também fazendo o uso do *Deep Learning*, Wen et al. (2018) propuseram um método usando CNNs para realizar um diagnóstico de falhas em diversas bases de dados, incluindo bases com dados de rolamentos, bombas centrífugas e bombas hidráulicas de pistão axial. Para usar a CNN, o método proposto transforma dados dos sinais do sistema de uma dimensão para imagens de duas dimensões. A fim de avaliar o algoritmo, os resultados foram comparados com outras arquiteturas de CNNs e uma SVM, sendo que uma arquitetura de CNN, baseada na LeNet-5 (LECUN et al., 1998), apresentou os melhores resultados.

No contexto automobilístico, Wang et al. (2014) usaram um método inteligente baseado em ruído para a identificação de falhas em motores. Usando as técnicas da transformada de *Hilbert-Huang* e máquina de suporte de vetor, foi possível criar um modelo eficaz para a classificação destas falhas. Para verificar a eficiência do método, a validação cruzada foi utilizada, o que garantiu um resultado confiável. Os autores ressaltam que o método utilizado pode ser empregado para outras aplicações de diagnóstico de falhas na engenharia.

Li, Cao e Yang (2018) utilizaram uma SVM para classificar múltiplas falhas em um sistema de geração elétrica, a partir de hidrogênio. Como se trata de um sistema com muitas variáveis, também foi utilizado o PCA para diminuir a dimensionalidade do problema. De acordo com os atores, os resultados foram de acordo com o esperado e suficientes para aumentar significantemente a segurança e confiabilidade do sistema.

3.8 Considerações finais

Conforme apresentado neste Capítulo, existem vários estudos com objetivo de diagnosticar falhas fazendo o uso de técnicas de AM nos mais diversos sistemas. No entanto, não foram encontrados trabalhos que consideram dados oriundos de módulos de rastreamento de veículos.

Alguns destes trabalhos fazem a extração de características de forma manual, e outros de forma automática. Desta forma, existem vantagens e desvantagens destas abordagens que também serão analisadas nesta pesquisa. Todo o desenvolvimento desta pesquisa está detalhado no Capítulo 4.

4 Abordagens

4.1 Considerações iniciais

Como apontado na Capítulo 1 desta dissertação, um dos objetivos deste trabalho é aplicar técnicas de AM para o diagnóstico de falhas. Esta aplicação foi divida em duas abordagens, uma utilizando conhecimento de um especialista do sistema para realizar a extração de características, e outra sem fazer o uso de qualquer conhecimento específico sobre a aplicação, trabalhando diretamente com dados crus. A abordagem que faz o uso do conhecimento do especialista é denominada "com informação" e a abordagem que não faz o uso desde conhecimento é denominada "sem informação".

Na abordagem "com informação", foram utilizadas as seguintes técnicas tradicionais de AM: *Random Forest*, *Naive Bayes*, SVM e MLP, enquanto na abordagem "sem informação", alguns modelos de CNNs, baseados na arquitetura da VGG-16 (SIMONYAN; ZISSERMAN, 2014) foram empregados.

O estudo de caso desta pesquisa é um problema de diagnóstico de falhas dos módulos de rastreamento da frota de veículos, onde várias falhas podem ocorrer. Todos os dados usados são reais e foram concedidos pela empresa DDMX[1].

Neste Capítulo estão descritos o problema abordado e as abordagens propostas para aplicação dos modelos de aprendizado de máquina.

4.2 Problema

O problema abordado nesta pesquisa consiste no diagnóstico de falhas de módulos rastreadores de frotas veiculares. Podem ocorrer diversas falhas em um módulo, causando um custo adicional para a empresa que gerencia estas frotas. Desta forma, é interessante diagnosticar falhas nestes módulos, a fim de minimizar os custos de manutenção, como o envio de técnicos e trocas de módulos.

Na Figura 18, pode-se observar o fluxo de dados da empresa DDMX. Na Figura 18, nota-se a existência de um seletor que faz a padronização inicial dos dados coletados pelos módulos. Isto é necessário, pois a empresa faz o uso de módulos de fabricantes diferentes. O coletor faz o direcionamento dos dados para tratamento e uso futuro no *core* da empresa, e armazena todos os dados coletados na coleção *positions* do BD da empresa. O *core* é responsável pelo processamento dos dados, assim ele lê os dados da coleção *positons* e gera

[1] http://ddmx.com.br

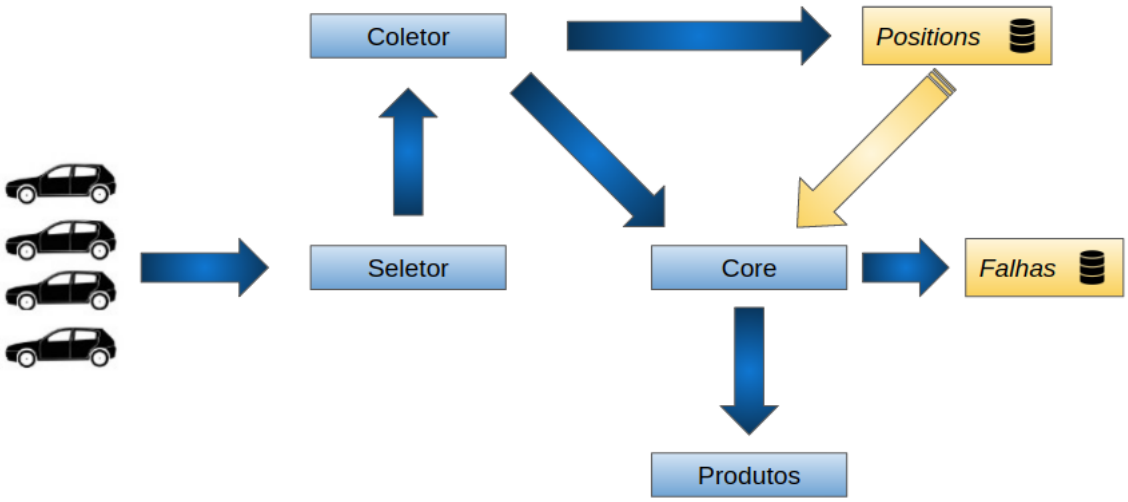


Figura 18 – Esquema de funcionamento da empresa.

um relatório contendo diversas informações dos módulos, e este relatório é armazenado na coleção *falhas* do BD da empresa.

Nestes dados existem sete falhas diferentes entre si já categorizadas, podendo existir mais falhas não identificadas pela empresa. Também já existe um sistema de diagnóstico dessas falhas, no entanto, ele é feito de forma simplória. Assim, a partir dos dados recolhidos pelos módulos de monitoramento, esta pesquisa pode apresentar um sistema para efetuar o diagnóstico de falhas aplicado aos módulos rastreadores da empresa de forma inteligente.

Nota-se que, estas falhas não estão distribuídas uniformemente no BD, para esta pesquisa foram disponibilizados 12586 registros. Por registro, entende-se todos os dados recolhidos de um determinado veículo no período de um dia. A primeira data contabilizada é 23/02/2018 e a última é 31/07/2018.

No BD disponibilizado existe uma coleção denominada *positions*. Na qual estão armazenados pontos contendo os dados recebidos dos módulos rastreadores da frota de veículos. Assim, existe uma quantidade alta de dados na coleção *positions*, já que um módulo envia em média 1116,67 pontos por dia. Constatou-se que estes pontos têm uma taxa de envio, e esta pode variar de acordo com a configuração e condições de uso do módulo, por exemplo, caso o módulo esteja em movimento a taxa de envio dos pontos aumenta.

Além da coleção *positions*, também há a coleção *falhas*, onde existe um relatório criado levando em consideração os pontos recebidos por um determinado veículo durante o período de um dia. Este relatório de falhas contém diversos dados originalmente usados para diagnosticar as falhas. Em cada relatório existem 74 atributos como variação de

ignição, valores máximos e mínimos da tensão na bateria e alternador, além de vários atributos usados para outras aplicações da empresa. Apesar destes relatórios conterem toda esta quantidade de atributos, apenas os campo de $falhas$, *data* e *serial* foram utilizados nesta pesquisa, pois é deste relatório que os rótulos de cada registro foram retirados. Para fazer a correlação entre as coleções *positions* e $falhas$, os campos de *data* e *serial* foram utilizados.

Cada ponto enviado pelo módulo contém os 62 atributos, sendo muitos destes descartados por um especialista do sistema, por diversos motivos, como por exemplo, dados que não são configurados em todos módulos e dados temporários usados para outras aplicações. OS atributos descartados são mostrados na Tabela 1 e os atributos mantidos são apresentados na Tabela 2.

Tabela 1 – Lista de atributos descartados de cada *Position*.

Atributo	**Descrição**
aceleração	Descartado pelo especialista do sistema.
ad_1	Descartado pelo especialista do sistema.
ad_2	Descartado pelo especialista do sistema.
ad_3	Descartado pelo especialista do sistema.
ad_4	Descartado pelo especialista do sistema.
Antena_desconectada	Descartado pelo especialista do sistema.
csq	Descartado pelo especialista do sistema.
Data	Descartado pelo especialista do sistema.
Direção	Descartado pelo especialista do sistema.
Angulo	Descartado pelo especialista do sistema.
Distância	Descartado pelo especialista do sistema.
Jamming	Descartado pelo especialista do sistema.
Distancia_real	Descartado pelo especialista do sistema.
Embreagem	Descartado pelo especialista do sistema.
Falha_antena	Descartado pelo especialista do sistema.
Freio	Descartado pelo especialista do sistema.
Fuzzy	Descartado pelo especialista do sistema.
GPS_moving	Descartado pelo especialista do sistema.
GPS_valido	Descartado pelo especialista do sistema.
hdop	Descartado pelo especialista do sistema.
Horimetro_motor	Descartado pelo especialista do sistema.
iccid	Descartado pelo especialista do sistema.
input_1	Descartado pelo especialista do sistema.
input_2	Descartado pelo especialista do sistema.

Tabela 1 – Lista de atributos descartados de cada *Position*.

Atributo	**Descrição**
input_3	Descartado pelo especialista do sistema.
input_4	Descartado pelo especialista do sistema.
input_5	Descartado pelo especialista do sistema.
input_6	Descartado pelo especialista do sistema.
input_7	Descartado pelo especialista do sistema.
input_8	Descartado pelo especialista do sistema.
Limpador_parabrisa	Descartado pelo especialista do sistema.
Motivo_envio	Descartado pelo especialista do sistema.
Motorista	Descartado pelo especialista do sistema.
Nivel_combustivel	Descartado pelo especialista do sistema.
Hodometro_telemetria	Descartado pelo especialista do sistema.
output_1	Descartado pelo especialista do sistema.
output_2	Descartado pelo especialista do sistema.
output_3	Descartado pelo especialista do sistema.
output_4	Descartado pelo especialista do sistema.
output_5	Descartado pelo especialista do sistema.
output_6	Descartado pelo especialista do sistema.
output_7	Descartado pelo especialista do sistema.
output_8	Descartado pelo especialista do sistema.
Panico	Descartado pelo especialista do sistema.
Passageiro	Descartado pelo especialista do sistema.
Pressao_oleo	Descartado pelo especialista do sistema.
Rota	Descartado pelo especialista do sistema.
RPM	Descartado pelo especialista do sistema.
rs232	Descartado pelo especialista do sistema.
Temp_motor	Descartado pelo especialista do sistema.
Tensão_alternador	Descartado pelo especialista do sistema.
Tipo	Descartado pelo especialista do sistema.

Tabela 2 – Lista de atributos mantidos de cada *Position*.

Atributo	**Descrição**
Bateria	Tensão na bateria do veículo.
Data_servidor	*String* com a data e hora registrada no servidor.
idx_memoria	Posição da memória que o ponto está armazenado.
Ignição	Valor binário que indica se a ignição está ativa.

Tabela 2 – Lista de atributos mantidos de cada *Position*.

Atributo	**Descrição**
Latitude	Latitude do veículo.
Longitude	Longitude do veículo.
Hodômetro	Indica quantos quilômetros o veículo já percorreu;
Serial	Identificador único que indica qual é o módulo.
Sinal_GPS	Indica se o sinal do GPS é válido.
Velocidade	Velocidade calculada pelo módulo.

Além da descrição dos dados inicialmente utilizados, também é necessário descrever cada uma das falhas que podem ser diagnosticadas. Para facilitar a identificação de cada falha, elas são referenciadas por números cardeais.

Sete falhas foram mapeadas, de acordo com os defeitos que a empresa já identificou ocorrência nos módulos. Para realizar este mapeamento, cada falha foi isolada de forma que sejam independentes entre si, ou seja, nada impede que existam falhas múltiplas em um mesmo módulo. Este mapeamento pode ser visto na Tabela 3.

Tabela 3 – Descrição de cada falha.

Falha	**Descrição**
Falha 1	Pulso mau configurado.
Falha 2	Hodômetro travado.
Falha 3	GPS travado.
Falha 4	Fio da ignição solto.
Falha 5	Acelerômetro defeituoso.
Falha 6	*Buffer* do módulo com problema.
Falha 7	GPS com saltos na localização.

Como os módulos enviam dados constantemente e em diferentes frequências de acordo com o comportamento de cada veículo, cada registro tem uma quantidade diferente de dados. Na Equação 4.1, P_k é o número de pontos que um módulo envia durante o período de um dia, e oito é o número de atributos por ponto (descartando o serial e a data), portanto, um determinado módulo K envia N_k total de atributos.

$$N_k = P_k \times 8 \tag{4.1}$$

A partir desses atributos, foi possível desenvolver modelos de classificação supervisionados para classificar qual falha um registro possui ou se o módulo está funcionando corretamente.

4.3 Abordagem com informação

Para esta abordagem, devido a particularidade dos dados usados, foi necessário desenvolver uma metodologia para a elaboração dos modelos de diagnóstico de falhas. Esta metodologia é detalhada na subseção 4.3.1.

4.3.1 Metodologia

Como dito na seção 3.3, existem diversos procedimentos para classificar dados. Para esta pequisa, devido a particularidade dos dados utilizados, uma metodologia foi proposta para melhor resolver o problema deste estudo de caso.

A Figura 19 mostra o fluxograma da metodologia, que foi proposto para este estudo de caso. Esse procedimento é simples e direto, contendo os fundamentos de cada processo de AM, e alguns detalhes específicos para o estudo de caso. Cada passo desta metodologia pode ser explicado de seguinte forma:

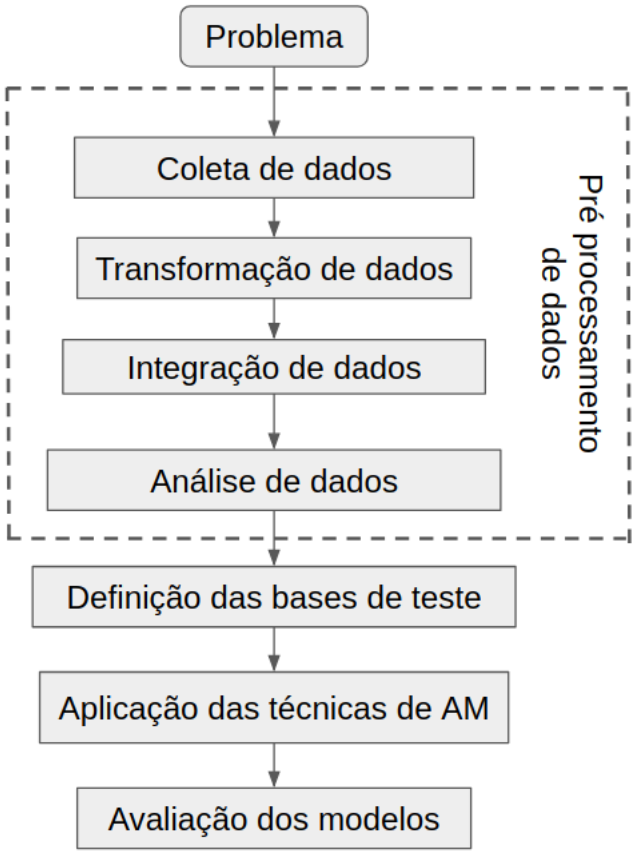


Figura 19 – Fluxograma do procedimento de classificação utilizado.

- **Problema**: Primeiramente, o problema deve ser descrito de forma simples para entende-lo. Também, deve-se saber qual(is) classe(s) deve(m) ser classificada(s), conforme detalhado na seção 4.2;

- **Coleta de dados**: Para a aplicação de qualquer técnica de AM, dados são necessários. Os dados devem ser coletados e padronizados, para que o método seja capaz de entender todos os dados de forma inteligente;
- **Transformação de dados**: Neste problema, é necessário realizar uma redução de dimensionalidade, detalhada na subseção 4.3.3. Para isto, atributos foram selecionados quanto a sua relevância na classificação das falhas e transformados conforme orientação do especialista do sistema;
- **Integração de dados**: Cada registro não estava inicialmente com o seu rótulo, ou classe indicada. Para isto, foi necessário criar uma função para relacionar duas coleções do banco de dados (BD). O processo é detalhado na subseção 4.3.4;
- **Análise de dados**: Esta análise é necessária para entender os dados e a partir deste entendimento realizar mudanças, para que as técnicas de AM tenham um resultado otimizado;
- **Definição das bases de teste**: A partir dos dados retirados, foram definidas diversas estruturas do mesmo conjunto de dados, cada uma com seu objetivo. Este processo é melhor descrito na subseção 5.2.1;
- **Aplicação das técnicas de AM**: Para cada uma das estruturas do conjunto de dados criadas foram separados os conjuntos de treinamento e teste, e aplicadas cada uma das técnicas de AM a serem analisadas nesta pesquisa;
- **Avaliação das técnicas**: Os resultados das técnicas pode ser avaliado por meio das métricas: precisão, TVP, medida F_1, e a matriz de confusão de cada um dos experimentos.

Nas próximas subseções, o procedimento executado nesta abordagem é detalhado dentro da metodologia proposta.

4.3.2 Coleta de dados

Para retirar os dados da empresa foram fornecidas credenciais de acesso, já que os dados são privados, e apenas cedidos para a pesquisa. Para realizar o acesso ao BD tanto como o restante da pesquisa, a distribuição *Anaconda* foi utilizada juntamente com a linguagem *Python* 3.7. No cenário da retirada de dados, as bibliotecas *PyMongo* 3.7.0, *NumPy* 1.15.1 e *Pandas* 0.23.4 foram utilizadas.

Para padronizar os dados, cada registro sofreu um tratamento no momento da sua retirada. Este tratamento consiste em operações de conversão de tipo de variáveis, já que algumas variáveis que estavam armazenadas no BD encontravam-se fora do padrão adotado pela linguagem de programação. Estas variáveis são: latitude, longitude, velocidade,

bateria e hodômetro. Por exemplo, a variável *odometro* inicialmente tem valores, como "1.254.051,32", o qual é transformado em "1254051.32", para que a linguagem reconheça este valor como um ponto flutuante.

A empresa forneceu um total de 142340 registros, no entanto, há muito mais registros sem falhas dos quais apresentam falhas, o que qualifica o BD como desbalanceado. Desta forma, a coleta de dados efetua um balanceamento por sub-amostragem no BD, coletando menos dados que apresentam falhas. A relação da quantidade de registros retirados e seu conjunto de falhas, pode ser visto na Tabela 4.

Tabela 4 – Quantidade de dados retirados do DB.

Identificação da falha	Quantidade de registros
Sem falha	5106
Falha 2	2170
Falha 3	1573
Falha 4	1702
Outras	2035
Total	12586

Pode-se verificar na Tabela 4, que algumas falhas aconteceram numa frequência significantemente menor. Portanto, as falhas que não apresentaram uma quantidade de registros maior que 3% do total de falhas, foram aglomeradas em um único conjunto denominado "Outras", representando todas as falhas que ocorrem em baixa frequência.

4.3.3 Transformação de dados

Como o processo de extração de características é um trabalho exaustivo e afeta significantemente o desempenho dos modelos (WEN et al., 2018), a transformação de dados usou o conhecimento fornecido por um especialista do sistema. A partir do conhecimento especializado, foi realizada uma redução de dimensionalidade nos dados. Os atributos iniciais eram séries temporais (tensão na bateria, latitude, etc.), permitindo a criação de 21 novos atributos artesanais, que foram criados a partir do conhecimento do especialista. Estes atributos podem ser vistos na Tabela 5. Assim, a transformação de dados converte os dados originais de duas dimensões para apenas uma dimensão.

A maioria destes atributos podem ser calculados de forma simples, e por esta razão não há uma fórmula ou algoritmo complexo para isto. O único atributo que necessitou de uma fórmula foi o valor RMS (*Root Mean Square*) da tensão na bateria. A fórmula para o cálculo pode ser vista na Equação 4.2 (BOYLESTAD, 2010), onde N é a quantidade de pontos no dia, e v_i é o valor de cada tensão na bateria enviada durante o dia.

$$V_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} v_i^2} \tag{4.2}$$

Tabela 5 – Descrição dos novos atributos.

Atributo	**Descrição**
Data	Data do registro
Serial	Código que identifica o módulo
Variação da longitude	Valor booleano que indica se a longitude variou
Variação da latitude	Valor booleano que indica se a latitude variou
Variação da ignição	Valor booleano que indica se a ignição variou
Variação do hodômetro	Valor booleano que indica se o hodômetro variou
Variação da velocidade	Valor booleano que indica se a velocidade variou
Tensão máxima na bateria	Ponto flutuante da tensão máxima registrada na bateria
Tensão mínima na bateria	Ponto flutuante da tensão mínima registrada na bateria
Tensão máxima na bateria com a ignição ligada	Ponto flutuante da tensão máxima registrada na bateria com a ignição ligada
Tensão máxima na bateria com a ignição desligada	Ponto flutuante da tensão máxima registrada na bateria com a ignição desligada
Tensão mínima na bateria com a ignição ligada	Ponto flutuante da tensão mínima registrada na bateria com a ignição ligada
Tensão mínima na bateria com a ignição desligada	Ponto flutuante da tensão mínima registrada na bateria com a ignição desligada
Variação da tensão na bateria	Ponto flutuante da tensão mínima na bateria subtraída da tensão máxima registrada na bateria
Valor RMS da tensão na bateria	Ponto flutuante do valor RMS do sinal da bateria registrado nesta data
Maior pulo positivo no índex de memória	Valor inteiro do maior pulo positivo registrado no índex de memória
Maior pulo negativo no índex de memória	Valor inteiro do maior pulo negativo registrado no índex de memória
Número de pulos no índex de memória	Valor inteiro do número de pulos registrados no índex de memória
Número de pontos com o GPS válido	Valor inteiro do número de pontos enviados pelo módulo com o GPS válido
Número de pontos com o GPS inválido	Valor inteiro do número de pontos enviados pelo módulo com o GPS inválido
Quantidade de pontos	Ponto flutuante da quantidade de pontos enviados pelo módulo nesta data
Quantidade de pulos no GPS	Ponto flutuante da quantidade de pulos registrados nos valores de longitude e latitude

A escolha de cada um destes atributos foi feita baseada na experiência no diagnóstico de falhas manual por funcionários da DDMX. Por exemplo, para verificar se o GPS do módulo está defeituoso, basta verificar se a velocidade do veículo variou sem ocorrer variação na latitude e na longitude, logo descartam-se todos os valores de velocidade, latitude e longitude, e usam-se apenas valores binários indicando se houve variação nessas variáveis.

4.3.4 Integração de dados

O rótulo de cada registro se encontra na coleção $falhas$, assim é necessário fazer uma relação entre as coleções *positions* e $falhas$. Para isto foi realizada uma busca no banco para cada registro coletado. Nesta busca, o objetivo foi encontrar o dado que representa o registro na coleção $falhas$.

Para realizar esta busca na coleção $falhas$, os dados de *serial* e *data* do registro foram utilizados a fim de encontrar o dado do mesmo módulo na mesma data, e armazenar o campo que representa as falhas como uma nova coluna no conjunto de dados.

Notou-se que o campo de data na coleção $falhas$ estava em um formato diferente deste mesmo campo no conjunto de dados. Então, este formato foi convertido de "MM DD, AAAA" para "DD/MM/AAAA", sendo DD o dia, MM o mês e AAAA o ano, para então realizar esta busca.

Desta forma, uma nova coluna foi gerada no conjunto de dados, indicando qual é a falha que este registro possui. A partir desta coluna foi feita uma verificação para identificar os registros que possuem falhas, criando uma nova coluna "$possui\_falhas$", contendo uma variável binária para indicar a existência de falhas no registro.

4.3.5 Análise de dados

Como o objetivo desta pesquisa foi avaliar os métodos de AM para o diagnóstico de falhas é possível e plausível que *outliers* sejam indicativos de falhas no sistema. Desta forma, não foi feita uma análise completa de *outliers*.

Boa parte dos atributos são binários, logo não necessitam de normalização. No entanto, atributos como a quantidade de pontos, dados referentes à tensão na bateria e pulos no índex de memória podem ser normalizados. Como não foi feita uma análise de *outliers*, uma normalização tradicional pode fazer com que os dados sejam suprimidos por algum valor alto demais, então é utilizada a biblioteca de preprocessamento do *Sk-learn* e a implementação do *RobustScaler* (PEDREGOSA et al., 2011) que segue a Equação 4.3. Nesta equação pode-se ver que ela usa o primeiro e terceiro quartis como limitantes e não os valores máximos e mínimos, assim esta normalização é mais robusta e mantém os *outliers*.

$$S_i = \frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)} \tag{4.3}$$

Como existem vários atributos parecidos, foi feito um mapa de calor mostrando a correlação de *Pearson* entre cada atributo. A ideia deste mapa é visualizar de forma simples, caso existam, atributos com uma correlação muito alta, e assim, podem ser considerados dados redundantes. Este mapa pode ser visto na Figura 20. Os campos mais vermelhos indicam uma correlação positiva alta, e os campos mais azulados indicam uma correlação negativa alta, enquanto que os campos brancos indicam a inexistência de relação entre os atributos.

Através deste mapa de calor foi possível identificar uma alta correlação (99,6315%) entre as variações de latitude e longitude, o que na maioria das projetos de AM poderia

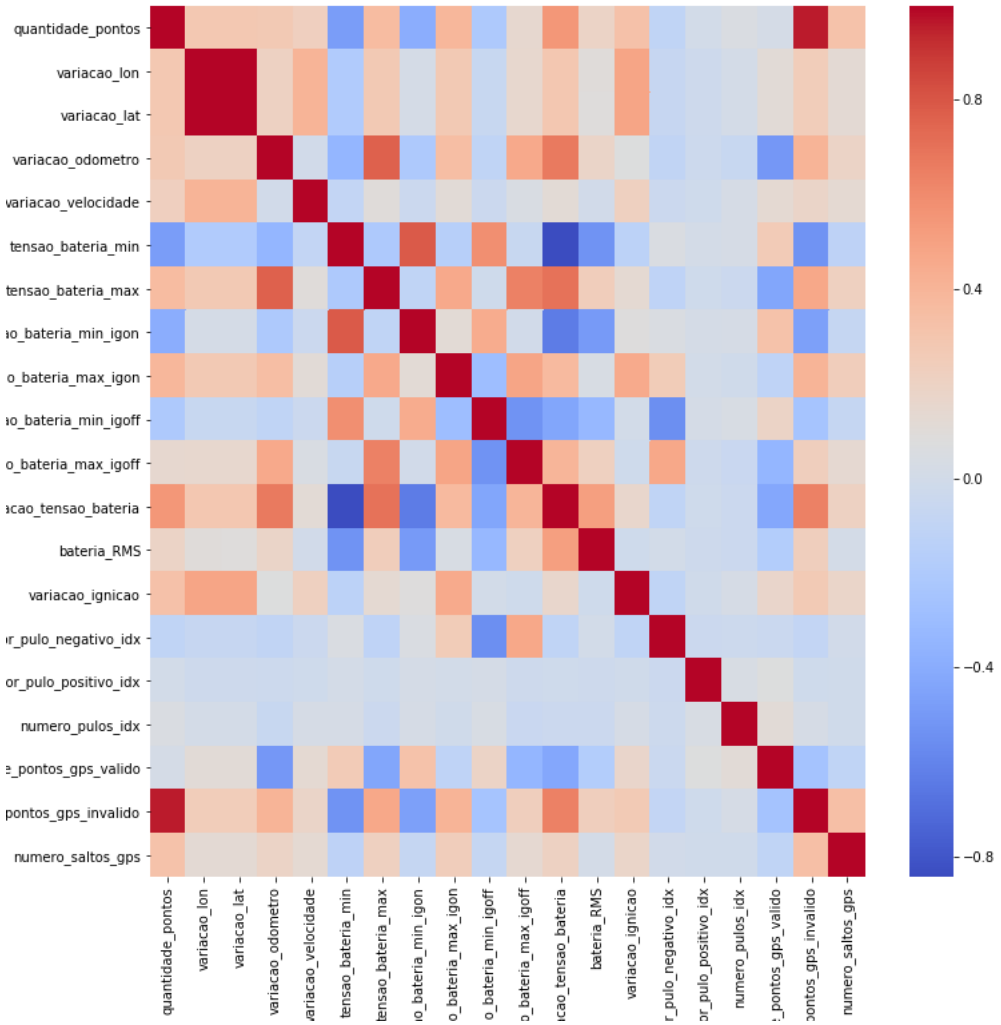


Figura 20 – Mapa de calor mostrando correlação entre atributos.

acarretar no descarte de um destes atributos. No entanto, como um dos objetivos dos modelos é identificar falhas no GPS do módulo, ambos atributos são úteis, apesar da alta correlação.

Também foi possível visualizar uma alta correlação (96,2769%) entre a quantidade de pontos e a quantidade de pontos com GPS inválido. Isto deve-se ao fato de que, a quantidade de pontos é a soma da quantidade de pontos com GPS válido, com a quantidade de pontos com o GPS inválido. Logo, devido a esta combinação linear, o atributo "quantidade de pontos" foi descartado. Desta forma, constatou-se que inicialmente existiam 22 atributos, no entanto a data, o serial e a quantidade de pontos foram descartados, restando 19 atributos.

Isto conclui o pré-processamento de dados, o qual foi fundamental para padronizar

os dados de entrada e garantir que os dados fossem organizados de forma que os métodos AM pudessem ser aplicados sem grandes dificuldades. Desta forma, os experimentos referentes a esta abordagem são detalhados na seção 5.2, e o restante deste Capítulo faz a descrição da abordagem "sem informação".

4.4 Abordagem sem informação

Nesta abordagem, os modelos de AM trabalham diretamente com os dados crus dos módulos rastreadores, sem realizar uma extração de características como na abordagem "com informação". Para isto, foram utilizados modelos de redes neurais profundas capazes de extrair automaticamente as características do conjunto de dados.

4.4.1 Metodologia

A metodologia adotada nesta abordagem é descrita no fluxograma da Figura 21, onde é possível notar que ainda existe um processo de pré-processamento dos dados, mas este mantém todo o conjunto original de dados sem criar novos atributos.

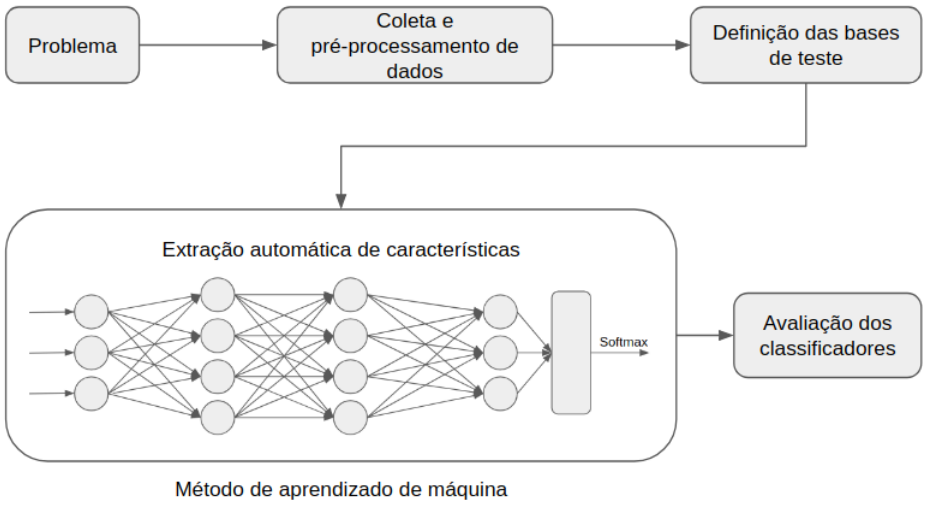


Figura 21 – Fluxograma de metodologia da abordagem sem informação.

Esta metodologia também é simples e direta. Cada etapa pode ser descrita da seguinte forma:

- **Problema**: apresentação, formulação e descrição do problema de forma simples, conforme detalhado na seção 4.2;
- **Coleta e pré-processamento de dados**: a coleta de dados é realizada da mesma forma descrita na subseção 4.3.2, e o pré-processamento é detalhado na subseção 4.4.2;

- **Definição das bases de teste**: a partir dos dados retirados são definidas estruturas do mesmo conjunto de dados, cada uma com seu objetivo. O processo será descrito na subseção 5.3.3;

- **Método de aprendizado de máquina**: descrição do planejamento de experimentos, treinamento dos modelos e apresentação dos resultados obtidos nesta abordagem. O processo é melhor descrito na subseção 5.3.1;

- **Avaliação dos classificadores**: os modelos obtidos podem ser avaliados e comparados considerando a precisão, TVP, medida F_1 e matrizes de confusão.

4.4.2 Coleta e pré-processamento de dados

Assim como descrito na subseção 4.3.2, no momento da retirada de dados, os mesmos foram coletados do DB e padronizados. Diferentemente da abordagem "com informação", desta vez os rótulos dos registros ficam em um diretório separado.

Como visto nas Tabelas 1 e 2, cada ponto enviado pelos módulos possui 62 atributos, dentre estes oito foram selecionados como úteis pelo especialista do sistema. No entanto, como o objetivo deste trabalho foi realizar a extração de características sem conhecimento do especialista, todos os dados coletados foram pré-processados, não apenas estes oito.

O primeiro passo deste pré-processamento consistiu em descartar os dados que não fazem sentido para a classificação, dados redundantes e dados nulos. Após isto, restaram 41 atributos por ponto enviado, que são: ad_1, ad_2, ad_3, ad_4, antena_desconectada, bateria, embreagem, falha_antena, freio, gps_valido, idx_memoria, ignicao, input_1, input_2, input_3, input_4, input_5, input_6, input_7, input_8, jamming, latitude, limpador_parabrisa, longitude, motor_ligado, nivel_combustivel, hôdometro, output_1, output_2, output_3, output_4, output_5, output_6, output_7, output_8, panico, pressao_oleo, sinal_gps, temp_motor, tensao_alternador e velocidade.

O segundo passo constituiu na normalização de todos os dados, novamente foi utilizada a normalização robusta (Equação 4.3), não suprimindo os dados pela presença de *outliers*.

Como a quantidade de atributos por registro é diferente, esta quantidade pode ser definida pela Equação 4.4, onde P_k é o número de pontos que um módulo envia durante o período de um dia, e 41 é o número de atributos por ponto, portanto um determinado módulo K envia N_k total de atributos.

$$N_k = P_k \times 41 \tag{4.4}$$

Como os registros possuem tamanho variável, não foi possível construir diretamente os modelos de AM neste conjunto de dados. Desta forma, foi necessário definir um tamanho padrão para todos os registros. Sabendo que a maior quantidade de pontos presentes em um registro é 6410, a média de pontos é 1116,67, e o desvio padrão é 1155,75, os registros com uma quantidade de dados menor foram completados com "0s", até possuírem a mesma quantidade de dados do maior registro. Desta forma, todos os registros têm quantidade total de atributos igual a 262810 (6410×41).

Outras abordagens para padronização da quantidade de dados poderiam ser executadas, como truncar a quantidade de dados dos registros com quantidade acima da média, e completar com "0s" os que estão abaixo da média, ou truncar a quantidade de dados de todos registros com mais dados que o menor registro. No entanto, estas abordagens foram testadas e consideradas inviáveis, devido a divergência dos modelos de AM.

Deste modo, os dados são tratados de forma que possam treinar uma CNN. Os experimentos envolvendo esta abordagem e as redes neurais utilizadas são descritos na seção 5.3.

4.5 Considerações finais

Neste Capítulo, o problema do estudo de caso desta pesquisa é descrito, juntamente com as abordagens propostas, para realizar o diagnóstico de falhas através de técnicas de AM. É possível notar que o processo de tratamento dos dados na abordagem "sem informação" é mais simples e rápido que o processo na abordagem "com informação", isto se dá principalmente devido a forma de extração de características.

No Capítulo 5, o planejamento dos experimentos realizados, juntamente com seus resultados e discussões, são apresentados.

5 Experimentos e discussões

5.1 Considerações iniciais

Após o tratamento dos dados, como descrito no Capítulo 4, é possível realizar o treinamento dos modelos de AM. Neste Capítulo, o planejamento dos experimentos em conjunto com seus respectivos resultados e discussões são apresentados.

5.2 Experimentos com informação

5.2.1 Definição das bases de teste

No intuito de melhorar a análise dos resultados e descobrir se alguma falha foi mapeada erroneamente, além de comparar a eficiência das técnicas em diferentes tarefas, foi realizada uma divisão no conjunto de dados. A partir do conjunto de dados original, duas estruturas do mesmo conjunto de dados foram definidas, uma apenas para detectar falhas (Estrutura 1), e uma para detectar e identificar falhas (Estrutura 2). Estas estruturas são descritas da seguinte maneira:

- **Estrutura 1**: criada para identificar se existem falhas. Contém 12586 registros, sendo 7480 apresentando falhas e 5106 sem falhas;
- **Estrutura 2**: criada para comparar e identificar dados sem falha e com falhas mistas, incluindo "outras". Contém 5106 registros sem falhas, 2170 com a falha 2, 1573 com a falha 3, 1702 com a falha 4 e 2035 com a falha "outras", totalizando 12586 registros.

5.2.2 Aplicação dos métodos

Nesta subseção, o planejamento dos experimentos é descrito, considerando o conjunto de dados referente ao problema abordado neste trabalho. Todos os modelos foram executados em um ambiente com as configurações detalhadas na Tabela 6, linguagem *Python* 3.6 e a biblioteca *Sk-Learn* 0.20.2.

Tabela 6 – Configurações do ambiente utilizado.

CPU	Intel i7-7700HQ 2,80GHz
Memória RAM	16Gb DDR4 2600Mhz
GPU	Nvidia GeForce GTX 1050Ti
Sistema Operacional	Linux Mint 19.01

Para executar os algoritmos, os conjuntos de dados foram separados em conjunto de treinamento, e conjunto de teste. Para cada experimento, 20% dos registros foram usados para testes, e 80% para treinamento, o que é uma prática comum na literatura (TAN et al., 2018).

Os experimentos foram planejados usando fatores de avaliação, e variáveis de resposta (JAIN, 2013). Como pode ser visto na Tabela 7, existem três fatores com 4, 2 e 2 níveis, respectivamente, onde os fatores correspondem às características dos experimento e os níveis aos possíveis valores. Todas as combinações de técnicas, conjunto de dados e amostragem são realizadas, totalizando 16 experimentos. As variáveis de resposta consideradas são: precisão, TVP, medida F_1 e também a matriz de confusão.

Tabela 7 – Planejamento de experimentos

Fatores	**Níveis**
Técnica	*Random Forest*, *Naive Bayes*, SVM, MLP
Amostragem	Sobre-amostragem, sub-amostragem
Estrutura	1, 2

É importante destacar que todos os modelos de *Random Forest* foram criados usando 100 estimadores, todos os modelos do tipo MLP empregaram arquitetura (5, 10, 5), lr (taxa de aprendizado) igual a $0,001$, e valor máximo de iterações igual a 200. Todas SVMs foram treinadas com o *kernel* rbf e foi realizada uma busca em grade para encontrar os parâmetros C e γ. A sobre-amostragem é feita utilizando o SMOTE e a sub-amostragem é realizada com o *RandomUndersampler*.

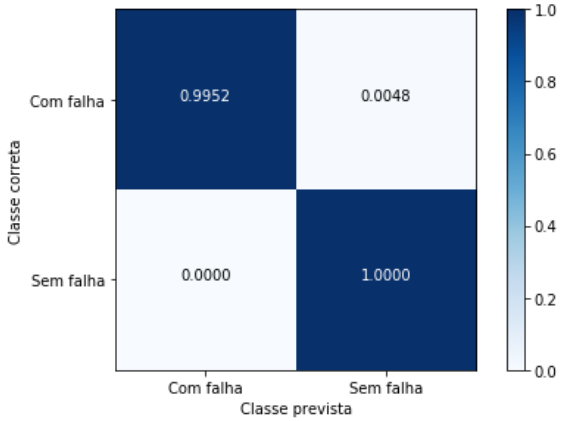
Cada resultado executado no conjunto de teste é mostrado na forma de uma tabela apresentando os valores obtidos para as métricas consideradas e uma matriz de confusão. Em seguida, é realizada a análise individual dos modelos, e uma análise geral da abordagem "com informação". Por fim, uma análise comparativa entre as duas abordagens é realizada na seção 5.4.

5.2.2.1 Estrutura 1

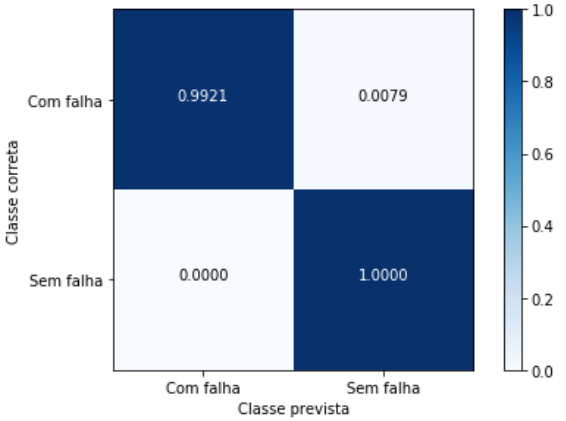
- ***Random Forest***: A Tabela 8 apresenta os valores obtidos para as métricas, considerando sub-amostragem e sobre-amostragem, respectivamente. E ainda, na mesma situação, as matrizes de confusão geradas podem ser observadas nas Figuras 22a e 22b. A *Random Forest* apresentou os melhores resultados quantitativos. Nota-se que a taxa de falso positivo foi maior que a taxa de falso negativo. Também, a diferença entre o modelo sub-amostrado e o sobre-amostrado é pequena, mas a sub-amostragem se demonstrou mais eficiente.

Tabela 8 – Relatório de classificação da *Random Forest* na estrutura 1 [%].

	Sub-amostragem			Sobre-amostragem		
	Precisão	TVP	F_1	Precisão	TVP	F_1
Sem falha	100,00	99,52	99,76	100,00	99,21	99,61
Com falha	99,50	100,00	99,75	99,47	100,00	99,73
Média	99,76	99,76	99,76	99,68	99,68	99,68



(a) Matriz de confusão da *Random Forest* para a estrutura 1 usando a sub-amostragem.



(b) Matriz de confusão da *Random Forest* para a estrutura 1 usando a sobre-amostragem.

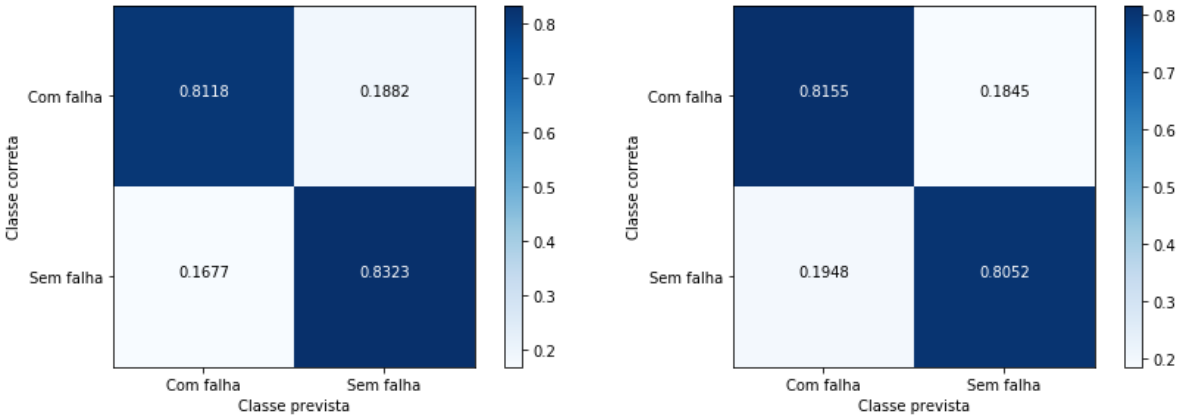
Figura 22 – Matrizes de confusão relativas a *Random Forest* para a estrutura 1.

- ***Naive Bayes***: A Tabela 9 apresenta os valores obtidos para as métricas, considerando sub-amostragem e sobre-amostragem, respectivamente. E ainda, na mesma situação, as matrizes de confusão geradas podem ser observadas nas Figuras 23a e 23b. O *Naive Bayes* obteve os piores resultados para a estrutura em questão. Com taxas de falsos positivos e falsos negativos próximas, o conjunto de treinamento foi submetido a classificação resultado em 82,78% na medida F_1, o que indicou um sub-ajuste nestes modelos. Além disso, o modelo sub-amostrado se demonstrou mais eficiente em relação ao sobre-amostrado.

Tabela 9 – Relatório de classificação da *Naive Bayes* na estrutura 1 [%].

	Sub-amostragem			Sobre-amostragem		
	Precisão	TVP	F_1	Precisão	TVP	F_1
Sem falha	83,58	81,18	82,36	74,00	81,55	77,59
Com falha	80,80	83,23	82,00	86,52	80,52	83,41
Média	82,22	82,18	82,19	81,45	80,94	81,06

- **SVM**: A Tabela 10 apresenta os valores obtidos para as métricas, considerando sub-amostragem e sobre-amostragem, respectivamente. E ainda, na mesma situação, as matrizes de confusão geradas podem ser observadas nas Figuras 24a e 24b. Os parâmetros utilizados foram $C = 1000$, $\gamma = 0.01$ e o *kernel* RBF. Depois da *Random*



(a) Matriz de confusão da *Naive Bayes* para a estrutura 1 usando a sub-amostragem.

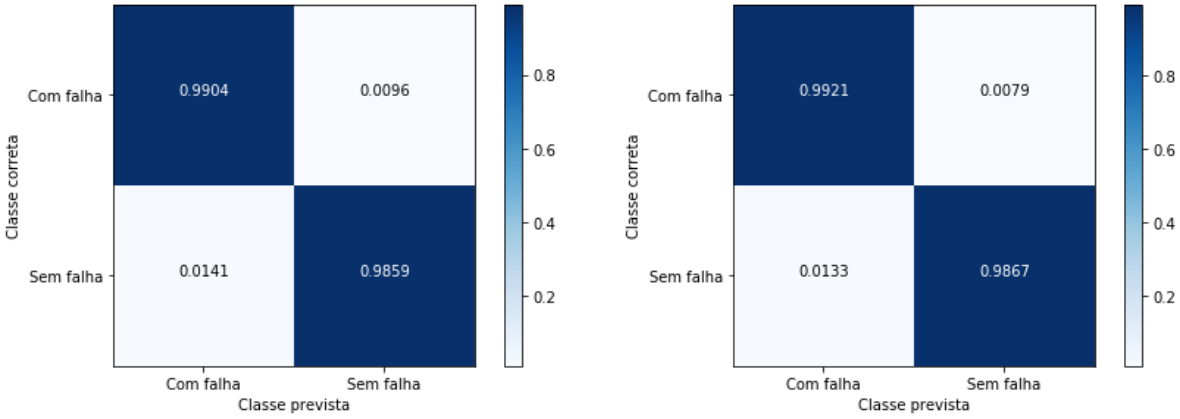
(b) Matriz de confusão da *Naive Bayes* para a estrutura 1 usando a sobre-amostragem.

Figura 23 – Matrizes de confusão relativas a *Naive Bayes* para a estrutura 1.

Forest, a melhor eficiência geral foi registrada pela SVM. Assim como o *Naive Bayes*, o modelo sub-amostrado obteve melhores resultados que o sobre-amostrado.

Tabela 10 – Relatório de classificação da SVM na estrutura 1 [%].

	Sub-amostragem			Sobre-amostragem		
	Precisão	TVP	F_1	Precisão	TVP	F_1
Sem falha	98,67	99,04	98,86	98,06	99,21	98,63
Com falha	98,99	98,59	98,79	98,67	99,06	99,06
Média	98,83	98,83	98,83	98,89	98,89	98,89



(a) Matriz de confusão da SVM para a estrutura 1 usando a sub-amostragem.

(b) Matriz de confusão da SVM para a estrutura 1 usando a sobre-amostragem.

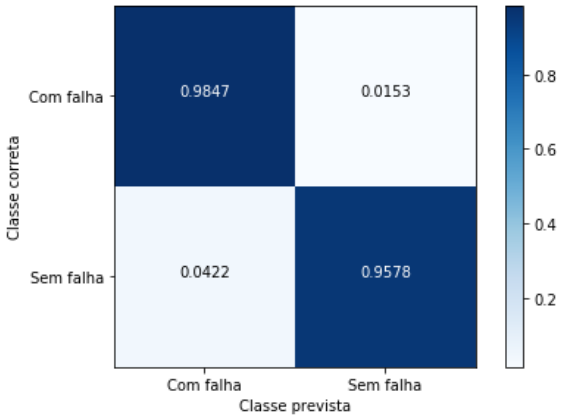
Figura 24 – Matrizes de confusão relativas a SVM para a estrutura 1.

- **MLP**: A Tabela 11 apresenta os valores obtidos para as métricas, considerando sub-amostragem e sobre-amostragem, respectivamente. E ainda, na mesma situação, as matrizes de confusão geradas podem ser observadas nas Figuras 25a e 25b. A MLP

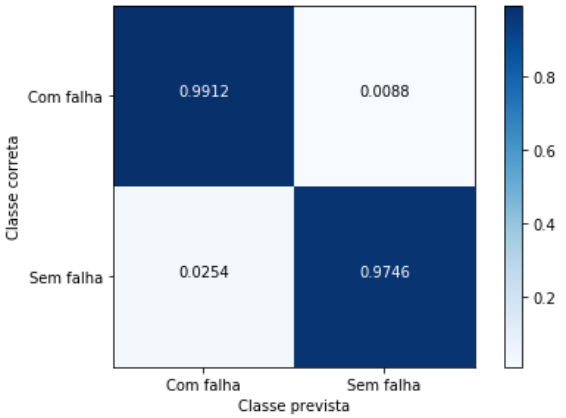
apresentou um desempenho semelhante a SVM. Apesar de ser uma rede neural simples, foi capaz de adaptar-se ao problema, o que demonstra a possibilidade de uma comparação com a abordagem "sem informação".

Tabela 11 – Relatório de classificação da MLP na estrutura 1 [%].

	Sub-amostragem			Sobre-amostragem		
	Precisão	TVP	F_1	Precisão	TVP	F_1
Sem falha	96,09	98,47	97,26	96,37	99,12	97,73
Com falha	98,35	95,78	97,05	99,39	97,46	98,42
Média	97,19	97,16	97,16	98,17	98,13	98,14



(a) Matriz de confusão da MLP para a estrutura 1 usando a sub-amostragem.



(b) Matriz de confusão da MLP para a estrutura 1 usando a sobre-amostragem.

Figura 25 – Matrizes de confusão relativas a MLP para a estrutura 1.

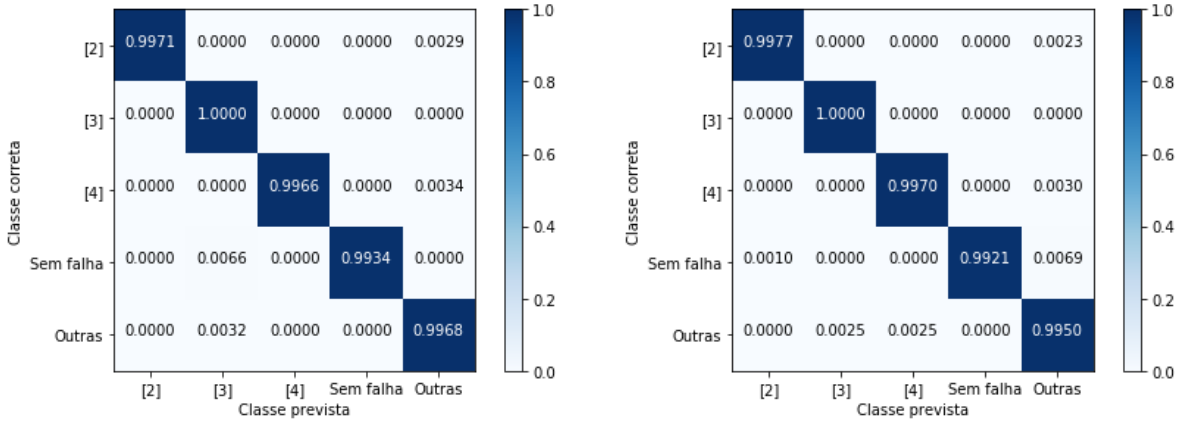
5.2.2.2 Estrutura 2

- ***Random Forest***: A Tabela 12 apresenta os valores obtidos para as métricas, considerando sub-amostragem e sobre-amostragem, respectivamente. E ainda, na mesma situação, as matrizes de confusão geradas podem ser observadas nas Figuras 26a e 26b. Na base de dados em questão, a *Random Forest* apresentou o melhor desempenho, com as maiores métricas obtidas.

- ***Naive Bayes***: A Tabela 13 apresenta os valores obtidos para as métricas, considerando sub-amostragem e sobre-amostragem, respectivamente. E ainda, na mesma situação, as matrizes de confusão geradas podem ser observadas nas Figuras 27a e 27b. Esta técnica apenas classifica bem a falha 2, havendo confusão entre todas as demais classes, especialmente a classe "Outras". Assim, como na estrutura 1, o *Naive Bayes* obteve métricas avaliativas baixas. Para conferir a possibilidade de um sub-ajuste, o conjunto de treinamento é avaliado obtendo uma medida F_1 de 82,19%, o que indica o sub-ajuste nestes classificadores.

Tabela 12 – Relatório de classificação da *Random Forest* na estrutura 2 [%].

	Sub-amostragem			Sobre-amostragem		
	Precisão	TVP	F_1	Precisão	TVP	F_1
Falha 2	100,00	99,71	99,85	99,77	99,77	99,77
Falha 3	99,08	100,00	99,54	99,69	100,00	99,84
Falha 4	100,00	99,66	99,83	99,70	99,70	99,70
Sem falha	100,00	99,34	99,67	100,00	99,21	99,61
Outras	99,37	99,68	99,53	97,80	99,50	98,54
Média	99,68	99,68	99,68	99,53	99,52	99,52



(a) Matriz de confusão da *Random Forest* para a estrutura 2 usando a sub-amostragem.

(b) Matriz de confusão da *Random Forest* para a estrutura 2 usando a sobre-amostragem.

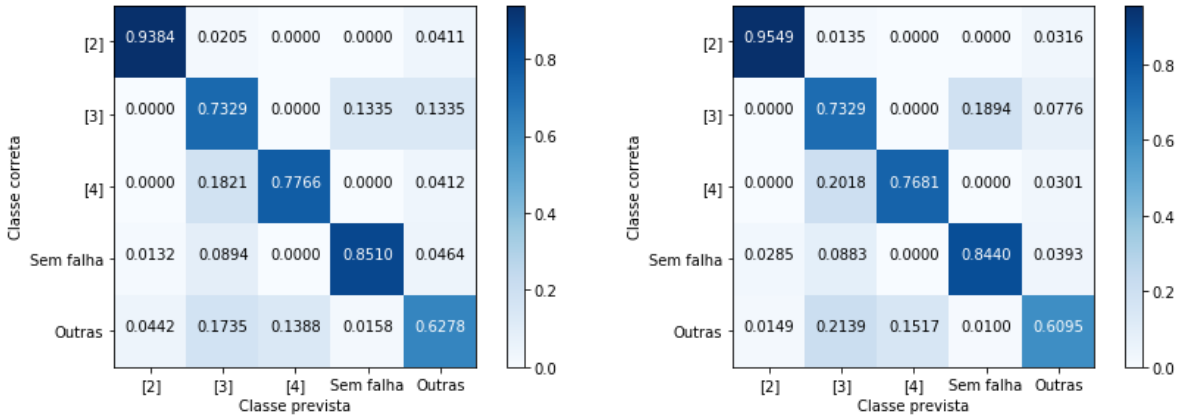
Figura 26 – Matrizes de confusão relativas a *Random Forest* para a estrutura 2.

Tabela 13 – Relatório de classificação da *Naive Bayes* na estrutura 2 [%].

	Sub-amostragem			Sobre-amostragem		
	Precisão	TVP	F_1	Precisão	TVP	F_1
Falha 2	94,67	93,84	94,26	92,36	95,49	93,90
Falha 3	62,43	73,29	67,43	48,66	73,29	58,49
Falha 4	83,70	77,66	80,57	80,70	76,81	78,70
Sem falha	84,26	85,10	84,68	92,97	84,40	88,48
Outras	70,57	62,78	66,44	73,35	60,95	66,58
Média	79,19	78,70	78,79	82,45	80,18	80,81

- **SVM**: A Tabela 14 apresenta os valores obtidos para as métricas, considerando sub-amostragem e sobre-amostragem, respectivamente. E ainda, na mesma situação, as matrizes de confusão geradas podem ser observadas nas Figuras 28a e 28b. Os parâmetros utilizados foram $C = 1000$, $\gamma = 0.01$ e o *kernel* RBF. Assim como a *Random Forest*, esta técnica apresenta bons resultados, demonstrando confusão entre as classes "Outras" e "Sem falhas".

- **MLP**: A Tabela 15 apresenta os valores obtidos para as métricas, considerando sub-amostragem e sobre-amostragem, respectivamente. E ainda, na mesma situação, as



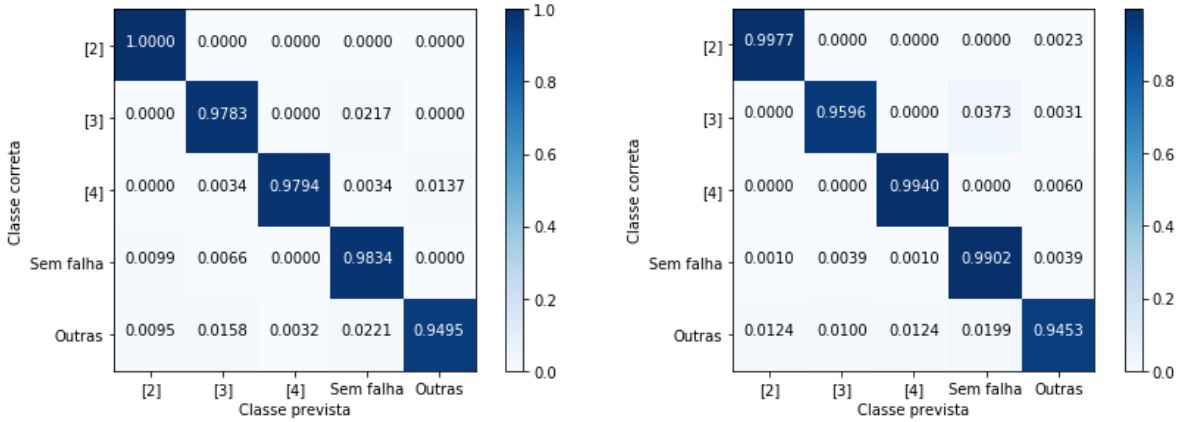
(a) Matriz de confusão da *Naive Bayes* para a estrutura 2 usando a sub-amostragem.

(b) Matriz de confusão da *Naive Bayes* para a estrutura 2 usando a sobre-amostragem.

Figura 27 – Matrizes de confusão relativas a *Naive Bayes* para a estrutura 2.

Tabela 14 – Relatório de classificação da SVM na estrutura 2 [%].

	Sub-amostragem			Sobre-amostragem		
	Precisão	TVP	F_1	Precisão	TVP	F_1
Falha 2	98,27	100,00	99,13	98,66	99,77	99,21
Falha 3	97,52	97,83	97,67	97,48	95,96	96,71
Falha 4	99,65	97,94	98,79	98,21	99,40	98,80
Sem falha	95,19	98,34	96,74	98,06	99,02	98,54
Outras	98,69	94,95	96,78	97,94	94,53	96,20
Média	97,87	97,84	97,84	98,09	98,09	98,08



(a) Matriz de confusão da SVM para a estrutura 2 usando a sub-amostragem.

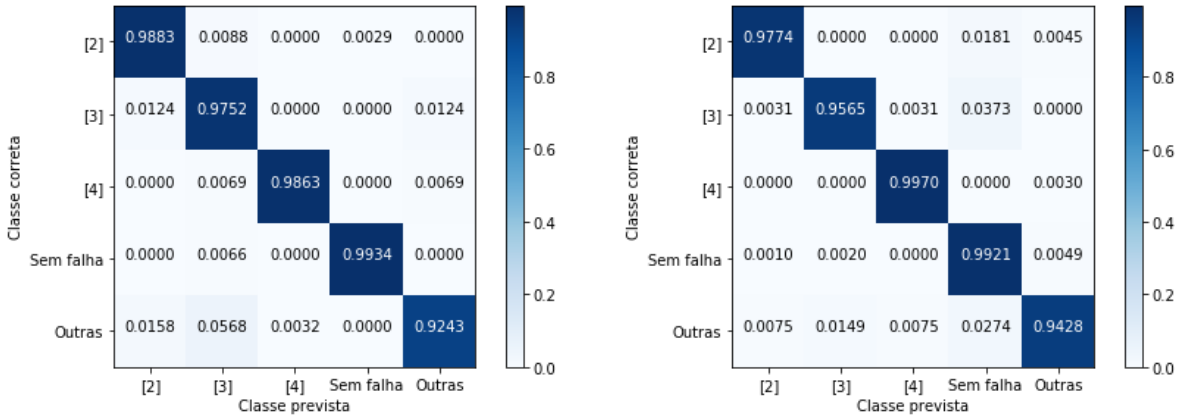
(b) Matriz de confusão da SVM para a estrutura 2 usando a sobre-amostragem.

Figura 28 – Matrizes de confusão relativas a SVM para a estrutura 2.

matrizes de confusão geradas podem ser observadas nas Figuras 29a e 29b. Mesmo com uma arquitetura simples, e cinco classes, a rede neural foi capaz de aprender e adaptar-se ao problema em questão, apesar de não ter obtido métricas tão elevadas quanto a *Random Forest*.

Tabela 15 – Relatório de classificação da MLP na estrutura 2 [%].

	Sub-amostragem			Sobre-amostragem		
	Precisão	TVP	F_1	Precisão	TVP	F_1
Falha 2	97,40	98,83	98,11	98,86	97,74	98,30
Falha 3	92,63	97,52	95,01	97,47	95,65	96,55
Falha 4	99,65	98,63	99,14	98,81	99,70	99,25
Sem falha	99,67	99,34	99,50	97,02	99,21	98,11
Outras	97,99	92,43	95,13	97,93	94,28	96,07
Média	97,39	97,33	97,33	97,78	97,78	97,77



(a) Matriz de confusão da MLP para a estrutura 2 usando a sub-amostragem.

(b) Matriz de confusão da MLP para a estrutura 2 usando a sobre-amostragem.

Figura 29 – Matrizes de confusão relativas a MLP para a estrutura 2.

5.2.3 Avaliação dos modelos

Nesta subseção, os modelos da abordagem "com informação" foram analisados e comparados. A técnica que melhor apresentou resultados quantitativos foi a *Random Forest*, a partir do modelo treinado por esta técnica, é possível identificar quais são os atributos mais importantes na classificação. Na Figura 30, é possível ver um gráfico de barras mostrando a importância relativa dos 15 atributos mais importantes para a classificação da *Random Forest*, usando a sub-amostragem na estrutura 1.

De forma análoga, a Figura 31, apresenta um gráfico de barras exibindo a importância relativa dos 15 atributos mais importantes para a classificação da *Random Forest*, usando a sub-amostragem na estrutura 2.

Exitem alguns atributos em comum na lista dos mais importantes para ambos objetivos. A variação na ignição, quantidade de saltos no GPS, variação do hodômetro e variação de velocidade são, em ambos casos, os atributos mais importantes.

Um outro aspecto que pode ser visualizado nos relatórios de classificação é a troca entre precisão e a TVP. É possível notar que na maioria dos relatórios, quando a precisão é alta a TVP é baixa, e quando a TVP é alta a precisão é baixa. Isto acontece devido a

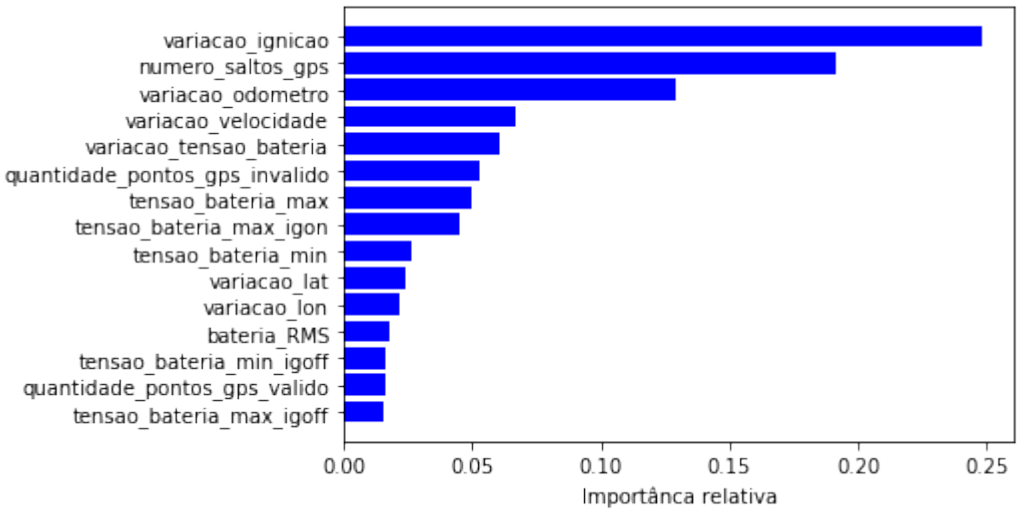


Figura 30 – Importância dos atributos principais para a estrutura 1.

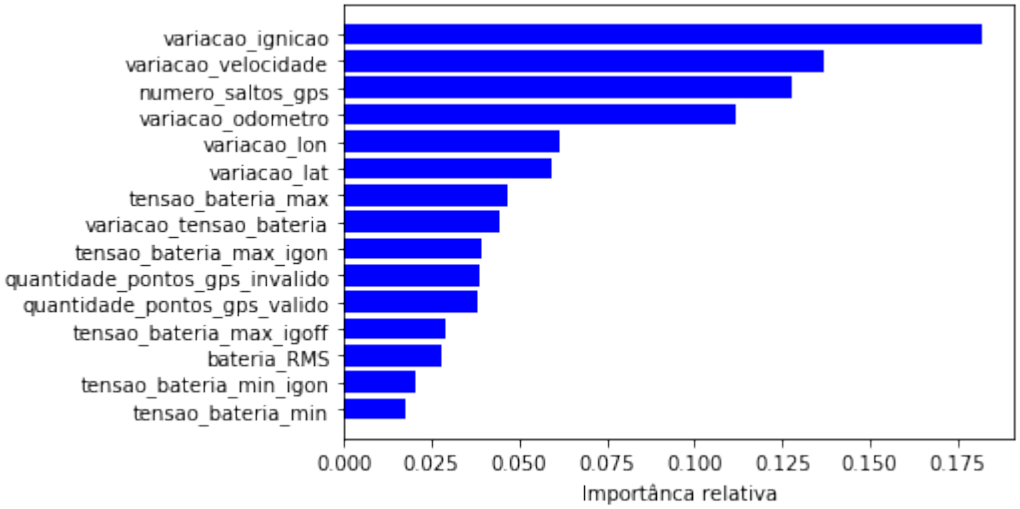


Figura 31 – Importância dos atributos principais para a estrutura 2.

natureza destas métricas avaliativas, enquanto a precisão é sensível à quantidade de falsos positivos, a TVP é sensível à falsos negativos.

Desta forma, as variáveis de resposta precisam de uma interpretação adequada. No estudo de caso em questão, é importante minimizar a quantidade de falsos positivos, já que isto implicaria em enviar um técnico para realizar uma manutenção desnecessária. Enquanto um falso positivo implica em receber uma reclamação de algum módulo com mau funcionamento. Assim, do ponto de vista financeiro, a TVP tem relevância maior que a precisão. Como critério de desempate, a medida F_1 é utilizada. Esta medida é a média harmônica entre a precisão e a TVP, logo apresenta-se como um critério avaliativo equilibrado, avaliando falsos positivos e falsos negativos.

Em toda a abordagem "com informação", constata-se que a média entre todas as precisões de um modelo e a média de todas as TVPs de um mesmo modelo, tende a anular a troca entre estas métricas. Também é possível analisar a quantidade de falsos positivos e falsos negativos diretamente na matriz de confusão, desta maneira é simples de analisar se o modelo tem uma tendência maior a classificar falsos positivos ou falsos negativos. Esta análise revela que, no geral, os modelos desta abordagem possuem uma tendência a classificar mais falsos negativos do que falsos positivos, o que é visto de maneira negativa do ponto de vista financeiro.

Há fatores em comum entre todas matrizes de confusão referentes à estrutura 2. Todos indicam uma confusão entre as falhas denominadas "Outras", e todas as demais falhas. Isto se dá devido a classe "Outras" ser um aglomerado de diversas falhas. Por consequência, o esperado é que haja confusão entre a classe "Outras" e as demais classes. Além disso, as métricas relativas a falha 2 sempre se mantêm acima de 95%, o que demonstra que esta é uma falha com maior facilidade de ser detectada pelas abordagens propostas.

Na estrutura 2, todas as técnicas, com a exceção da *Random Forest*, apresentaram uma performance superior com a sobre-amostragem. Isto indica que a *Random Forest* foi capaz de aprender com uma quantidade menor de informação.

Para melhor visualizar os resultados apresentados na subseção 5.2.2, foi feita a Tabela 16, onde todos experimentos podem ser comparados segundo algumas das métricas utilizadas. Por "US", "OS" e "Prec", entende-se respectivamente sub-amostragem, sobre-amostragem e precisão.

Tabela 16 – Métricas de avaliação para modelos da abordagem "com informação".

		Random Forest			Naive Bayes			SVM			MLP		
		Prec	TVP	F_1	Prec	TVP	F_1	Prec	TVP	F_1	Prec	TVP	F_1
E1	US	**99,76**	**99,76**	**99,76**	82,22	82,18	82,19	98,83	98,83	98,83	97,19	97,16	97,16
	OS	99,68	99,68	99,68	81,45	80,94	81,06	98,89	98,89	98,89	98,17	98,13	98,14
E2	US	**99,68**	**99,68**	**99,68**	79,19	78,70	78,79	97,87	97,84	97,84	97,39	97,33	97,33
	OS	99,53	99,53	99,53	82,45	80,18	80,81	98,09	98,09	98,08	97,78	97,78	97,77

Ainda na Tabela 16, nota-se que, a sub-amostragem apresenta melhores resultados em ambas estruturas de experimentação. Além disso, entre as quatro técnicas utilizadas, o *Random Forest* se destaca, seguida da SVM em todos os experimentos. O *Naive Bayes* tem o seu desempenho baixo em relação as demais técnicas, principalmente porque ele assume total independência entre os atributos, o que não acontece como é demonstrado na Figura 20.

Além das métricas de avaliação para as técnicas de AM, o tempo de treinamento e avaliação dos modelos também foi calculado. Estes dados podem ser vistos na Tabela 17, sendo que todos estes dados estão em ms.

Tabela 17 – Tempo de treinamento e avaliação dos modelos.

		Random Forest [ms]		*Naive Bayes* [ms]		SVM [ms]		MLP [ms]	
		Treinamento	Teste	Treinamento	Teste	Treinamento	Teste	Treinamento	Teste
E1	US	776,0	317,0	41,6	312,0	269000,0	494,0	3060,0	358,0
	OS	998,0	204,0	12,2	311,0	565000,0	538,0	3270,0	353,0
E2	US	528,0	314,0	79,0	324,0	100200,0	550,0	543,0	285,0
	OS	1212,0	383,0	114,0	474,0	865000,0	940,0	2510,0	270,0

Na Tabela 17, pode-se observar que o tempo de treinamento é geralmente maior que o teste, exceto para o *Naive Bayes*. Isto se dá devido a forma de treinamento desta técnica, que é naturalmente estatística, e requer uma quantidade de operações menor que qualquer outra das três técnicas.

É importante observar que a SVM tem um tempo de treinamento substancialmente maior que as outras técnicas, isto se dá, principalmente, devido a busca em grade realizada para encontrar os parâmetros usados.

O tempo de teste refere-se a avaliação de todo o conjunto de teste, logo pode-se calcular o tempo de avaliação médio de uma amostra dividindo o tempo total de teste pela quantidade de amostras. Este tempo é, no pior dos casos, $0,37ms$, o que pode ser considerado uma resposta em tempo real para a aplicação em questão. Levando em consideração que o treinamento dos modelos é apenas realizado uma vez, um tempo de treinamento não inviabiliza qualquer um dos modelos da abordagem "com informação".

A abordagem "com informação" se demonstra robusta, e apesar da necessidade de conhecimento especializado do sistema foi eficaz e apresentou uma avaliação positiva. Na seção 4.4, a abordagem "sem informação" será descrita juntamente com sua avaliação.

5.3 Experimentos sem informação

5.3.1 Método de aprendizado de máquina

Nesta subseção, o planejamento dos experimentos e as arquiteturas da redes neurais utilizadas são descritas. Todos os experimentos foram executados utilizando um ambiente com as mesmas configurações descritas na subseção 5.2.2, adicionando as seguintes bibliotecas: $Tensorflow$-gpu 1.12 e $Keras$ 2.2.4.

Assim, como na abordagem "com informação", para executar os algoritmos, os conjuntos de dados foram separados em conjunto de treinamento e conjunto de teste. Para cada experimento, 20% dos registros foram usados para teste, e 80% para treinamento.

Os fatores de avaliação, e os níveis usados no planejamento destes experimentos podem ser vistos na Tabela 18; são apenas dois fatores com dois níveis em cada. Todas as combinações de fatores são realizadas, totalizando 4 experimentos, um para cada es-

trutura. As variáveis de resposta consideradas foram: precisão, TVP, medida F_1 e matriz de confusão.

Tabela 18 – Planejamento de experimentos na abordagem sem informação.

Fatores	**Níveis**
Objetivo	Detectar falhas, detectar e identificar falhas
Seleção de atributos	Com seleção de atributos, Sem seleção de atributos

A técnica de AM utilizada nesta abordagem foi a CNN, da forma descrita na subseção 5.3.2, devido à alta capacidade de adaptação e generalização que esta rede possui.

A partir de cada estrutura definida, foi executado um experimento nas condições descritas nesta seção. Para cada experimento foram disponibilizados os resultados de teste e treinamento para a melhor discussão e análise dos modelos.

5.3.2 Arquitetura das CNNs

Uma CNN possui vários parâmetros que são de difícil otimização. Como o objetivo desta abordagem é avaliar a extração automática de dados, não houve um ajuste fino dos parâmetros da rede neural, sendo esta análise uma premissa para trabalhos futuros.

Os dados das estruturas 1 e 2 tem dimensões (6410, 41), e os dados das estruturas 3 e 4 tem dimensões (6410, 8). Isto se deve à seleção de atributos, implicando na necessidade de duas CNNs, uma para as estruturas 1 e 2, e outra para as estruturas 3 e 4. Desta forma, foram projetadas duas arquiteturas de CNN, ambas baseadas na arquitetura da VGG-16 (SIMONYAN; ZISSERMAN, 2014).

Na Tabela 19, está descrita toda a arquitetura da CNN usada nas estruturas 1 e 2. As siglas e abreviações CC, *Pool*, *Drop*, DC e SM significam respectivamente, Camada Convolucional, camada de sub-amostragem, camada *Dropout*, camada densamente conectada e camada com ativação *Softmax*. Na Tabela 20, é descrita arquitetura usada nas estruturas 3 e 4. As mudanças realizadas foram devido à dimensionalidade dos dados de entrada.

Na última camada, em cada uma das arquiteturas, o número n corresponde à quantidade de classes do problema envolvido. Ou seja, duas classes para as estruturas 1 e 3, e cinco para as estruturas 2 e 4. As camadas de sub-amostragem são do tipo *Average-Pooling*, em virtude dos dados coletados.

Como otimizador foi utilizado o Nadam (RUDER, 2016), que é o Adam juntamente com o momento de Nesterov, com taxa de aprendizado igual a $0,0001$. O tamanho da *batch size* usado foi um, devido aos custos computacionais atrelados. Os modelos foram

Tabela 19 – Arquitetura da CNN para as bases sem seleção de atributos.

Nome da camada	**Descrição da camada**
CC-1	128 filtros, (3, 3), ReLU
CC-2	128 filtros, (3, 3), ReLU
Pool-1	*Avg-Pooling*, (9, 2)
Drop-1	*Dropout*, 10%
CC-3	256 filtros, (3, 3), ReLU
CC-4	256 filtros, (3, 3), ReLU
Pool-2	*Avg-Pooling*, (7, 2)
Drop-2	*Dropout*, 10%
CC-5	512 filtros, (3, 3), ReLU
CC-6	512 filtros, (3, 3), ReLU
Pool-2	*Avg-Pooling*, (5, 1)
Drop-3	*Dropout*, 10%
CC-7	768 filtros, (3, 3), ReLU
CC-8	768 filtros, (3, 3), ReLU
Pool-2	*Avg-Pooling*, (3, 1)
Drop-4	*Dropout*, 10%
Flatten-1	Achatamento
DC-1	Densamente conectada, 4096 neurônios, ReLU
Drop-5	*Dropout*, 60%
DC-2	Densamente conectada, 768 neurônios, ReLU
Drop-6	*Dropout*, 20%
SM-1	Densamente conectada, n neurônios, *Softmax*

treinados durante 300 épocas, com possibilidade de interromper o treinamento caso a função de custo não sofra mais alterações que justifiquem o treinamento.

5.3.3 Definição das bases de teste

Assim como na abordagem "com informação", foram definidas estruturas do conjunto de dados original para cada experimento, sendo que nesta abordagem foram utilizados os mesmos registros que foram utilizados na abordagem "com informação", tanto no conjunto de treinamento, como o conjunto de teste. No total, quatro estruturas foram definidas, sendo que, além destas estruturas possuírem objetivos diferentes, também trabalham com quantidade de dados diferentes. Duas usam dados sem nenhuma seleção de atributos, e outras duas bases selecionam apenas oito atributos por ponto enviado. Estes oito são os selecionados como úteis pelo especialista na subseção 4.3.3. Ainda, devido ao desempenho superior da sub-amostragem na abordagem "com informação", e dificuldade de criar dados sintéticos com uma quantidade elevada de atributos, é realizada uma sub-amostragem para definir estas bases de teste, selecionando os dados de forma aleatória. Desta forma, foram definidas quatro estruturas, descritas da seguinte forma:

Tabela 20 – Arquitetura da CNN para as bases com seleção de atributos.

Nome da camada	**Descrição da camada**
CC-1	128 filtros, (4, 4), ReLU
CC-2	128 filtros, (4, 4), ReLU
Pool-1	*Avg-Pooling*, (9, 1)
Drop-1	*Dropout*, 10%
CC-3	256 filtros, (3, 3), ReLU
CC-4	256 filtros, (3, 3), ReLU
Pool-2	*Avg-Pooling*, (7, 1)
Drop-2	*Dropout*, 10%
CC-5	512 filtros, (3, 3), ReLU
CC-6	512 filtros, (3, 3), ReLU
Pool-2	*Avg-Pooling*, (5, 1)
Drop-3	*Dropout*, 10%
CC-7	768 filtros, (3, 3), ReLU
CC-8	768 filtros, (3, 3), ReLU
Pool-2	*Avg-Pooling*, (3, 1)
Drop-4	*Dropout*, 10%
Flatten-1	Achatamento
DC-1	Densamente conectada, 4096 neurônios, ReLU
Drop-5	*Dropout*, 60%
DC-2	Densamente conectada, 768 neurônios, ReLU
Drop-6	*Dropout*, 20%
SM-1	Densamente conectada, n neurônios, *Softmax*

- **Estrutura 1**: feita apenas para detectar falhas, sem seleção de atributos. Contém 10212 registros, sendo 5106 apresentando falhas e 5106 sem falhas. Cada registro possui 262810 atributos;

- **Estrutura 2**: feita para detectar e identificar falhas, sem seleção de atributos. Contém 7865 registros, sendo 1573 com a falha 2, 1573 com falha 3, 1573 com a falha 4 e 1573 com a falha "Outras". Cada registro possui 262810 atributos;

- **Estrutura 3**: feita apenas para detectar falhas, com seleção de atributos. Contém 10212 registros, sendo 5106 apresentando falhas e 5106 sem falhas. Cada registro possui 51280 atributos;

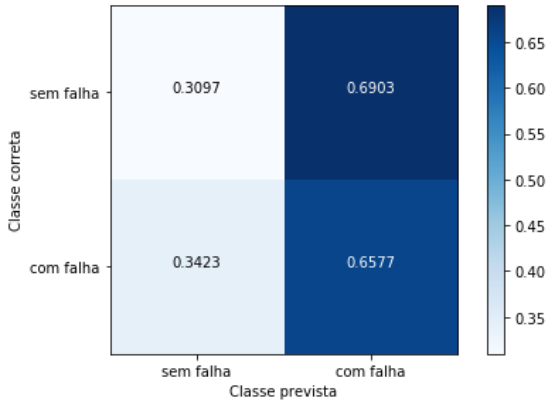
- **Estrutura 4**: feita para detectar e identificar falhas, com seleção de atributos. Contém 7865 registros, sendo 1573 com a falha 2, 1573 com falha 3, 1573 com a falha 4 e 1573 com a falha "Outras". Cada registro possui 51280 atributos.

5.3.3.1 Experimento 1

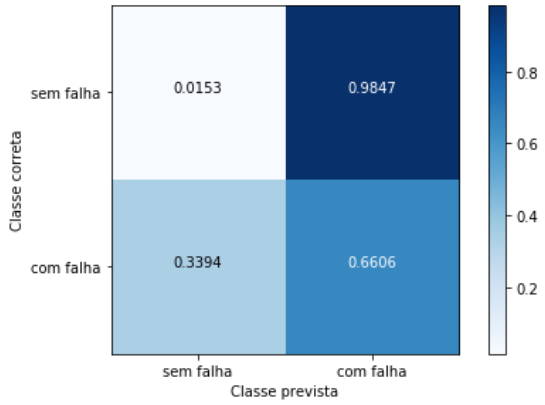
O experimento 1 consiste no treinamento e teste da CNN, com os objetivos descrito da estrutura 1, ou seja, dados sem seleção de atributos, modelo apenas para detecção de falhas. As métricas deste experimento usando os conjuntos de treinamento e teste podem ser vistas na Tabela 21 e suas respectivas matrizes de confusão podem ser vistas na Figura 32.

Tabela 21 – Relatório de classificação do experimento 1, para o conjunto de treinamento e conjunto de teste [%].

	Treinamento			Teste		
	Precisão	TVP	F_1	Precisão	TVP	F_1
Sem falha	47,18	30,97	37,39	04,52	01,53	02,28
Com falha	49,10	65,77	56,22	38,96	66,06	49,01
Média	48,15	48,48	46,87	21,31	32,99	25,07



(a) Matriz de confusão da CNN para o experimento 1.



(b) Matriz de confusão da CNN para o experimento 1.

Figura 32 – Matrizes de confusão relativas ao experimento 1.

Com métricas de treinamento próximas de 50%, este modelo apresenta sub-ajuste. No conjunto de teste, as métricas são ainda mais baixas, confirmando a que não houve aprendizado.

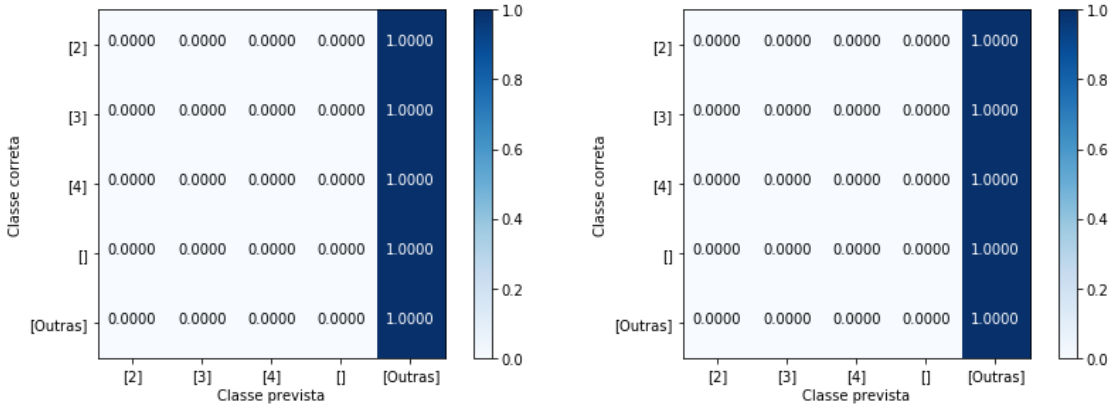
5.3.3.2 Experimento 2

O experimento 2 consiste no treinamento e teste da CNN, com os objetivos descrito da estrutura 2, ou seja, dados sem seleção de atributos, modelo para detecção e identificação de falhas. As métricas deste experimento usando os conjuntos de treinamento e teste podem ser vistas na Tabela 22 e suas respectivas matrizes de confusão podem ser vistas na Figura 33.

Para um problema envolvendo cinco classes, TVP próxima de 20%, e precisão abaixo de 5% nos conjuntos de treinamento e teste, demonstra novamente um sub-ajuste.

Tabela 22 – Relatório de classificação do experimento 2, para o conjunto de treinamento e conjunto de teste [%].

	Treinamento			Teste		
	Precisão	TVP	F_1	Precisão	TVP	F_1
Falha 2	0,00	0,00	0,00	0,00	0,00	0,00
Falha 3	0,00	0,00	0,00	0,00	0,00	0,00
Falha 4	0,00	0,00	0,00	0,00	0,00	0,00
Sem falha	0,00	0,00	0,00	0,00	0,00	0,00
Outras	20,15	100,00	33,54	19,96	100,00	33,28
Média	4,06	20,15	6,76	3,98	19,96	6,64



(a) Matriz de confusão da CNN para o experimento 2.

(b) Matriz de confusão da CNN para o experimento 2.

Figura 33 – Matrizes de confusão relativas ao experimento 2.

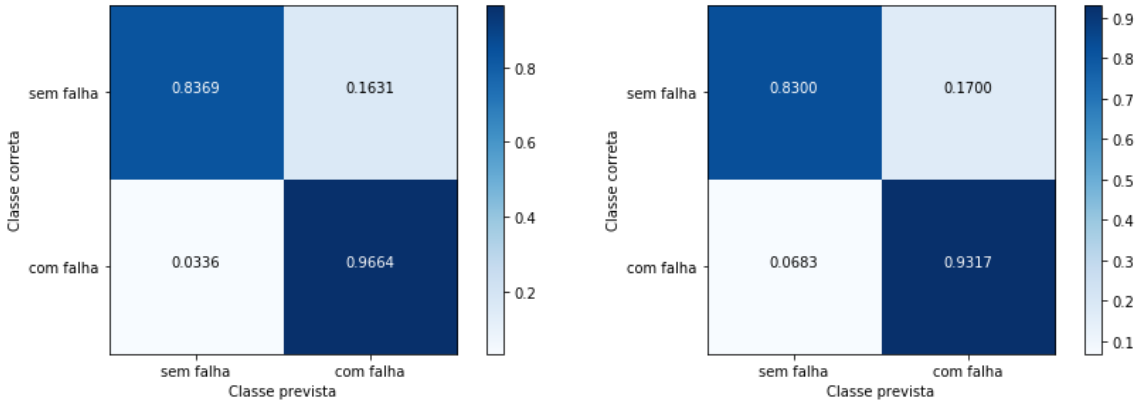
Todos registros foram avaliados para as classe “Outras”, confirmando a ineficiência no aprendizado deste modelo.

5.3.3.3 Experimento 3

O experimento 3 consiste no treinamento e teste da CNN, com os objetivos descrito da estrutura 3, ou seja, dados com seleção de atributos, modelo apenas para detecção de falhas. As métricas deste experimento usando os conjuntos de treinamento e teste podem ser vistas na Tabela 23 e suas respectivas matrizes de confusão podem ser vistas na Figura 34.

Tabela 23 – Relatório de classificação do experimento 3, para o conjunto de treinamento e conjunto de teste [%].

	Treinamento			Teste		
	Precisão	TVP	F_1	Precisão	TVP	F_1
Sem falha	96,10	83,69	89,47	92,74	83,00	87,60
Com falha	85,71	96,64	90,85	83,91	93,17	88,30
Média	90,87	90,21	90,16	88,43	87,96	87,94



(a) Matriz de confusão da CNN para o experimento 3.

(b) Matriz de confusão da CNN para o experimento 3.

Figura 34 – Matrizes de confusão relativas ao experimento 3.

Quantitativamente foi o melhor resultado desta abordagem, o modelo foi capaz de extrair características e classificar os dados. Todavia, é possível notar que existe um sub-ajuste. A quantidade de falsos positivos foi maior que a quantidade de falsos negativos, o que não é o ideal do ponto de vista financeiro.

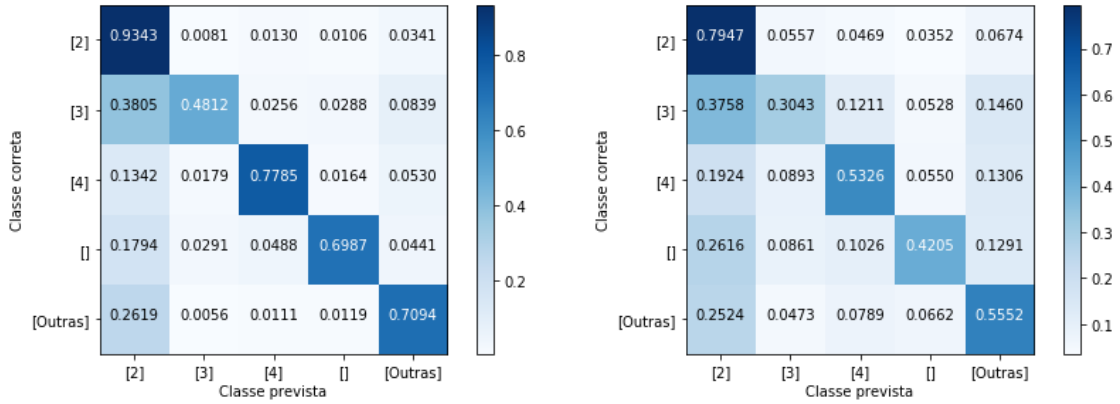
5.3.3.4 Experimento 4

O experimento 4 consiste no treinamento e teste da CNN, com os objetivos descrito da estrutura 4, ou seja, dados com seleção de atributos, modelo para detecção e identificação de falhas. As métricas deste experimento usando os conjuntos de treinamento e teste podem ser vistas na Tabela 24 e suas respectivas matrizes de confusão podem ser vistas na Figura 35.

Tabela 24 – Relatório de classificação do experimento 4, para o conjunto de treinamento e conjunto de teste [%].

	Treinamento			Teste		
	Precisão	TVP	F_1	Precisão	TVP	F_1
Falha 2	48,85	93,43	64,16	44,65	79,47	57,17
Falha 3	88,66	48,12	62,38	53,26	30,43	38,74
Falha 4	88,95	77,85	83,03	58,27	53,26	55,66
Sem falha	91,26	69,87	79,14	65,80	42,05	51,31
Outras	76,68	70,94	73,70	54,49	55,52	55,00
Média	79,06	72,00	72,58	54,98	52,57	51,55

Conseguindo também extrair características, este modelo foi capaz de identificar falhas. No entanto suas métricas avaliativas apontam um sob-ajuste em todas as classes. Além disso, a falha 2 demonstrou-se mais fácil de ser identificada, com TVP mais elevada, mas baixa precisão.



(a) Matriz de confusão da CNN para o experimento 4.

(b) Matriz de confusão da CNN para o experimento 4.

Figura 35 – Matrizes de confusão relativas ao experimento 4.

5.3.4 Avaliação dos classificadores

A partir dos relatórios de classificação presentes na subseção 5.3.1, a Tabela 25 tem como objetivo visualizar e comparar todos modelos da abordagem "sem informação", sendo que, SA significa seleção de atributos.

Tabela 25 – Métricas de avaliação para modelos da abordagem "sem informação".

	Detecção de falhas						Detecção e identificação de falhas					
	Treinamento			Teste			Treinamento			Teste		
	Prec	TVP	F_1	Prec	TVP	F_1	Prec	TVP	F_1	Prec	TVP	F_1
Sem SA	48,15	48,48	46,87	21,31	32,99	25,07	4,06	20,15	6,76	3,98	19,96	6,64
Com SA	**90,87**	**90,21**	**90,16**	**88,43**	**87,96**	**87,94**	**79,06**	**72,00**	**72,58**	**54,98**	**52,57**	**51,55**

É possível observar a importância da seleção de atributos para o processo de aprendizado do modelo. Na detecção de falhas e na detecção e identificação de falhas há, respectivamente duas, e cinco classes. Desta forma, pode-se dizer que o modelo não foi capaz de extrair características sem a seleção de atributos, concluindo que todos modelos sem a seleção de atributos sofrem de sub-ajuste.

A quantidade de atributos por registro nos modelos, com a seleção de atributos, é mais de cinco vezes menor, e ainda, a maioria dos dados presentes no conjunto de dados sem a seleção de atributos é zero, o que torna o processo de extração de características mais difícil. Já com a seleção de atributos, boa quantidade dos dados é diferente de zero, com a exceção do dados finais de cada registro.

As métricas avaliativas dos modelos com seleção de atributos demonstram que houve aprendizado, e extração automática de características, sendo a performance do modelo que apenas detecta falhas superior, devido a maior facilidade do problema proposto. A diferença de 2,25% e 19,43% entre as TVPs dos conjuntos de treinamento e teste mostram uma variância possivelmente evitável, com uma arquitetura mais regularizadora, ou

seja, uma arquitetura que vise evitar o sobre-ajuste.

Assim como na abordagem "com informação", a troca entre precisão e TVP é nítida nos relatórios de classificação. No entanto, nesta abordagem a diferença entre estas métricas é maior, acarretando uma avaliação negativa do ponto de vista financeiro. Isto pode ser justificado pelo sub-ajuste dos modelos propostos.

Os modelos sem seleção de atributos foram treinados por 50 épocas, e não apresentaram qualquer sinal de aprendizado que justificasse treinar estes modelos por mais tempo. Já os modelos com seleção de atributos foram treinados por 200 épocas, apresentando os melhores resultados nas épocas 137 e 157. Cada época tem período de treinamento próximo de $3000s$ para a detecção de falhas, e $2000s$ para a detecção e identificação de falhas.

O tempo de avaliação pode ser visualizado na Tabela 26. No pior caso, o tempo para a avaliação de um registro é $14{,}3ms$, o que não impossibilita a aplicação destes modelos na prática.

Tabela 26 – Tempo de avaliação dos modelos da abordagem "sem informação".

Modelo	Tempo[s]
Detecção de falhas, sem seleção de atributos	29,2
Detecção e identificação de falhas, sem seleção de atributos	19,7
Detecção de falhas, com seleção de atributos	26,4
Detecção e identificação de falhas, sem seleção de atributos	14,8

5.4 Comparação final

A partir das inferências realizadas na subseção 5.2.3, foi possível inferir que a extração de características realizada manualmente se demonstrou eficaz e obteve resultados bons. Com fundamento na subseção 5.3.4, também é possível dizer que a extração de características automática foi factível, no entanto esta obteve resultados inferiores à abordagem "com informação".

A Tabela 27, apresenta uma comparação final entre as abordagens, comparando apenas as maiores métricas, usando a TVP já que esta foi definida como variável de resposta de maior importância. Nesta tabela, observa-se que a abordagem "com informação" obteve uma TVP 9,55% maior que a abordagem "sem informação", o que significa uma performance superior.

Tabela 27 – Comparação final entre as abordagens, usando a TVP. [%]

Objetivo	Com informação	Sem informação
Detecção de falhas	**99,76**	87,96
Detecção e identificação de falhas	**99,68**	52,57

Ainda, como as CNNs da abordagem “sem informação” não são otimizadas e também podem ser coletados mais dados para o treinamento, estes resultados podem ser melhorados com um ajustes na arquitetura, parâmetros da rede neural e mudanças no pré-processamento de dados.

5.5 Considerações finais

Neste Capítulo, foi possível ver que a extração de características, embora muitas vezes seja trabalhosa e faz necessário um conhecimento avançado do sistema em questão, pode apresentar resultados melhores. No Capítulo 6, as conclusões e os trabalhos futuros serão expostos.

6 Conclusão e trabalhos futuros

Neste trabalho, foi apresentada a aplicação de um conjunto de técnicas de aprendizado de máquina no diagnóstico de falhas de módulos de rastreamento de frota de veículos. Para tal fim, duas abordagens foram propostas, uma fazendo o uso do conhecimento de um especialista do sistema para realizar uma extração de características manual e outra abordagem empregando redes neurais profundas para a extração de características automática.

Para a análise da abordagem "com informação", que considera o conhecimento do especialista, foram consideradas as técnicas tradicionais de AM: *Random Forest*, *Naive Bayes*, SVM e MLP. Para a aplicação dos modelos foi realizado um pré-processamento e a extração de características da base de dados, além do planejamento de experimentos para definição dos testes e posterior análise de resultados. Foram treinados e testados um total de 16 modelos fazendo uso de dados reais relativos aos módulos rastreadores de frotas de veículos. Estes modelos, utilizando a *Random Forest* no conjunto de dados sub-amostrado foi capaz de alcançar 99,79% e 99,68% na TVP para detectar falhas e detectar e identificar falhas.

Na abordagem "sem informação", são construídas CNNs para realizar o diagnóstico de falhas. Também foi necessário um pré-processamento dos dados e planejamento de experimentos, mas todo o processo de extração de características nesta abordagem é realizado pela rede neural de forma automática. Quatro modelos foram treinados e testados usando dados crus apenas com o pré-processamento descrito na seção 4.4. Estes modelos obtiveram 87,96% e 52,57% na TVP para detectar falhas e detectar e identificar falhas, ambos modelos realizando a seleção de atributos para uma limpeza prévia dos dados.

A partir da análise realizada é importante observar que, apesar da maioria dos modelos mais atuais de diagnóstico de falhas usarem abordagens com extração de características automáticas, isto não necessariamente se aplica a todo tipo de sistema. Nos dados referentes aos módulos rastreadores de frotas veiculares, a extração de características manual se revelou competente. No entanto, para realizá-la é necessário conhecimento do funcionamento dos módulos em questão.

A principal contribuição deste trabalho envolve a definição de duas abordagens para construção de modelos de AM capazes de diagnosticar falhas em dados oriundos de módulos rastreadores de frotas veiculares. Uma destas abordagens realizando a extração de características de forma manual, através do conhecimento de um especialista do sistema, e outra abordagem efetuando a extração de características de forma automática. Além disso, este trabalho realiza o detalhamento do funcionamento de cada modelo, e uma

análise qualitativa comparando cada modelo e sua respectiva abordagem.

Todos os modelos propostos não são capazes de descobrir falhas que a empresa não mapeou, portanto quaisquer falhas que não tenham sido mapeadas seriam classificadas erroneamente como uma das falhas conhecidas. Além disso, a extração de características é feita com um conhecimento especializado do sistema, de tal forma que é possível melhorá-la. Os modelos da abordagem "sem informação" podem ser otimizados, mudando a arquitetura das CNNs e outros parâmetros. Outros modelos como a LSTM, podem ser utilizados para realizar esta classificação. Desta forma, desenvolver modelos capazes de descobrir novas falhas, aprimorar a extração de características da abordagem "com informação", otimizar as CNNs propostas na abordagem "sem informação" e desenvolver modelos usando técnicas diferentes são propostas promissoras para trabalhos futuros.

Referências

ALBUQUERQUE, R. W. d. *Monitoramento da cobertura do solo no entorno de hidrelétricas utilizando o classificador SVM (Support Vector Machines).* Tese (Doutorado) — Universidade de São Paulo, 2012. Citado na página 41.

ANDRYCHOWICZ, M. et al. Learning to learn by gradient descent by gradient descent. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2016. p. 3981–3989. Citado na página 49.

AZEVEDO, A. I. R. L.; SANTOS, M. F. Kdd, semma and crisp-dm: a parallel overview. *IADS-DM*, 2008. Citado na página 36.

BASILIO, J. C.; CARVALHO, L. K.; MOREIRA, M. V. Diagnose de falhas em sistemas a eventos discretos modelados por autômatos finitos. *Revista Controle & Automaçao*, v. 21, n. 5, p. 510–533, 2010. Citado na página 26.

BATISTA, G. E.; PRATI, R. C.; MONARD, M. C. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, ACM, v. 6, n. 1, p. 20–29, 2004. Citado na página 52.

BEARD, R. V. *Failure accomodation in linear systems through self-reorganization.* Tese (Doutorado) — Massachusetts Institute of Technology, 1971. Citado na página 24.

BENGIO, Y.; LECUN, Y. et al. Scaling learning algorithms towards ai. *Large-scale kernel machines*, v. 34, n. 5, p. 1–41, 2007. Citado na página 46.

BISHOP, C. M. *Pattern recognition and machine learning*. [S.l.]: springer, 2006. Citado 2 vezes nas páginas 33 e 35.

BOYLESTAD, R. L. *Introductory Circuit Analysis*. [S.l.]: Prentice Hall Press, 2010. Citado na página 64.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Citado na página 40.

BREIMAN, L. et al. *Classification and regression trees*. [S.l.]: CRC press, 1984. Citado na página 38.

BURGES, C. J. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, Springer, v. 2, n. 2, p. 121–167, 1998. Citado 2 vezes nas páginas 41 e 43.

BUSSAB, W. O; morettin, pedro. a. *Estatística básica*, v. 4, 2010. Citado na página 44.

CABASINO, M. P.; GIUA, A.; SEATZU, C. Fault detection for discrete event systems using petri nets with unobservable transitions. *Automatica*, Elsevier, v. 46, n. 9, p. 1531–1539, 2010. Citado na página 26.

CAMILO, C. O.; SILVA, J. C. d. Mineração de dados: Conceitos, tarefas, métodos e ferramentas. *Universidade Federal de Goiás (UFC)*, 2009. Citado na página 36.

CAMPOS, T. E. de. *Técnicas de seleçao de caracterısticas com aplicaçoes em reconhecimento de faces*. Tese (Doutorado) — Master's thesis, Universidade de Sa Paulo, 2001. Citado na página 35.

CHAPELLE, O.; SCHOLKOPF, B.; ZIEN, A. *Semi-Supervised Learning*. [S.l.]: The MIT Press, 2006. (Adaptive Computation and Machine Learning). Citado na página 36.

CHARNIAK, E. Bayesian networks without tears. *AI magazine*, v. 12, n. 4, p. 50, 1991. Citado na página 44.

CHAWLA, N. V. et al. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, v. 16, p. 321–357, 2002. Citado na página 52.

CHEN, J.; PATTON, R. J. *Robust model-based fault diagnosis for dynamic systems*. [S.l.]: Springer Science & Business Media, 2012. v. 3. Citado 4 vezes nas páginas 11, 23, 24 e 25.

CHESTER, D.; LAMB, D.; DHURJATI, P. Rule-based computer alarm analysis in chemical process plants. In: MICRO-DELCON 84, IEEE. *Proceedings of the Seventh Annual Conference on Computer Technology*. [S.l.], 1984. p. 22–29. Citado na página 30.

CHURCHLAND, P. S.; SEJNOWSKI, T. J.; POGGIO, T. A. *The computational brain*. [S.l.]: MIT press, 2016. Citado na página 46.

CUPERTINO, T. H. *Machine learning via dynamical processes on complex networks*. Tese (Doutorado) — Universidade de São Paulo, 2014. Citado na página 35.

DAI, X.; GAO, Z. From model, signal to knowledge: A data-driven perspective of fault detection and diagnosis. *IEEE Transactions on Industrial Informatics*, IEEE, v. 9, n. 4, p. 2226–2238, 2013. Citado 2 vezes nas páginas 30 e 32.

DING, S. X. *Model-based fault diagnosis techniques: design schemes, algorithms, and tools*. [S.l.]: Springer Science & Business Media, 2008. Citado na página 23.

DÖHLER, M.; MEVEL, L. Subspace-based fault detection robust to changes in the noise covariances. *Automatica*, Elsevier, v. 49, n. 9, p. 2734–2743, 2013. Citado na página 25.

DOTOLI, M. et al. On-line fault detection in discrete event systems by petri nets and integer linear programming. *Automatica*, Elsevier, v. 45, n. 11, p. 2665–2672, 2009. Citado na página 26.

DUBATH, P. et al. Random forest automated supervised classification of hipparcos periodic variable stars. *Monthly Notices of the Royal Astronomical Society*, Blackwell Publishing Ltd Oxford, UK, v. 414, n. 3, p. 2602–2617, 2011. Citado na página 40.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, v. 12, n. Jul, p. 2121–2159, 2011. Citado na página 49.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern classification*. [S.l.]: John Wiley & Sons, 2012. Citado 3 vezes nas páginas 33, 35 e 50.

DUNIA, R.; QIN, S. J. Joint diagnosis of process and sensor faults using principal component analysis. *Control Engineering Practice*, Elsevier, v. 6, n. 4, p. 457–469, 1998. Citado na página 31.

ELNOKITY, O. et al. Ann based sensor faults detection, isolation, and reading estimates–sfdire: applied in a nuclear process. *Annals of Nuclear Energy*, Elsevier, v. 49, p. 131–142, 2012. Citado na página 31.

FACELI, K. et al. Inteligência artificial: Uma abordagem de aprendizado de máquina. *Rio de Janeiro: LTC*, v. 2, p. 192, 2011. Citado 2 vezes nas páginas 51 e 52.

FRANK, P. M.; DING, X. Survey of robust residual generation and evaluation methods in observer-based fault detection systems. *Journal of process control*, Elsevier, v. 7, n. 6, p. 403–424, 1997. Citado na página 24.

GALLAGHER, C.; MADDEN, M. G.; D'ARCY, B. A bayesian classification approach to improving performance for a real-world sales forecasting application. In: IEEE. *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*. [S.l.], 2015. p. 475–480. Citado na página 44.

GAN, M.; WANG, C. et al. Construction of hierarchical diagnosis network based on deep learning and its application in the fault pattern recognition of rolling element bearings. *Mechanical Systems and Signal Processing*, Elsevier, v. 72, p. 92–104, 2016. Citado na página 53.

GAO, Z.; CECATI, C.; DING, S. X. A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches. *IEEE Transactions on Industrial Electronics*, IEEE, v. 62, n. 6, p. 3757–3767, 2015. Citado 6 vezes nas páginas 19, 23, 24, 25, 26 e 27.

GAO, Z.; CECATI, C.; DING, S. X. A survey of fault diagnosis and fault-tolerant techniques—part ii: fault diagnosis with knowledge-based and hybrid/active approaches. *IEEE Transactions on Industrial Electronics*, 2015. Citado 7 vezes nas páginas 11, 24, 28, 30, 31, 32 e 33.

GARCIA, E. A.; FRANK, P. Deterministic nonlinear observer-based approaches to fault diagnosis: a survey. *control engineering practice*, Elsevier, v. 5, n. 5, p. 663–670, 1997. Citado na página 25.

GERTLER, J. J. Survey of model-based failure detection and isolation in complex plants. *IEEE Control systems magazine*, IEEE, v. 8, n. 6, p. 3–11, 1988. Citado na página 24.

GONÇALVES, L. S. M. *Categorizaçao em text mining*. Tese (Doutorado) — Universidade de São Paulo, 2002. Citado na página 44.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. `http://www.deeplearningbook.org`. Citado 2 vezes nas páginas 46 e 49.

GOPAKUMAR, P.; REDDY, M. J. B.; MOHANTA, D. K. Adaptive fault identification and classification methodology for smart power grids using synchronous phasor angle measurements. *IET Generation, Transmission & Distribution*, IET, v. 9, n. 2, p. 133–145, 2015. Citado na página 53.

GRUS, J. *Data Science do zero: Primeiras regras com o Python*. [S.l.]: Alta Books Editora, 2018. Citado na página 44.

GUO, J. et al. Fault diagnosis of hybrid systems using particle filter based hybrid estimation algorithm. *CHEMICAL ENGINEERING*, v. 33, 2013. Citado na página 26.

HALL, M. et al. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, ACM, v. 11, n. 1, p. 10–18, 2009. Citado na página 39.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. Overview of supervised learning. In: *The elements of statistical learning*. [S.l.]: Springer, 2009. p. 9–41. Citado na página 35.

HAYKIN, S. *Redes neurais: princípios e prática*. [S.l.]: Bookman Editora, 2007. Citado 3 vezes nas páginas 11, 45 e 46.

HE, Y.; MENDIS, G. J.; WEI, J. Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism. *IEEE Transactions on Smart Grid*, IEEE, v. 8, n. 5, p. 2505–2516, 2017. Citado na página 35.

HESSINE, M. B.; JOUINI, H.; CHEBBI, S. Fault detection and classification approaches in transmission lines using artificial neural networks. In: IEEE. *Mediterranean Electrotechnical Conference (MELECON), 2014 17th IEEE*. [S.l.], 2014. p. 515–519. Citado na página 53.

HINTON, G. E. Learning multiple layers of representation. *Trends in cognitive sciences*, Elsevier, v. 11, n. 10, p. 428–434, 2007. Citado na página 46.

HLAWATSCH, F.; AUGER, F. *Time-frequency analysis*. [S.l.]: John Wiley & Sons, 2013. Citado na página 28.

HOFMANN, T.; SCHÖLKOPF, B.; SMOLA, A. J. Kernel methods in machine learning. *The annals of statistics*, JSTOR, p. 1171–1220, 2008. Citado na página 42.

HONG, L.; DHUPIA, J. S. A time domain approach to diagnose gearbox fault based on measured vibration signals. *Journal of Sound and Vibration*, Elsevier, v. 333, n. 7, p. 2164–2180, 2014. Citado na página 27.

HSU, C.-W. et al. A practical guide to support vector classification. Taipei, Taiwan, 2003. Citado na página 43.

HSU, C.-W.; LIN, C.-J. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, IEEE, v. 13, n. 2, p. 415–425, 2002. Citado na página 43.

HUNT, E. B.; MARIN, J.; STONE, P. J. Experiments in induction. Academic press, 1966. Citado na página 38.

HUSON, D. *SVMs and Kernel Functions*. 2007. Disponível em: <http://ab.inf.uni-tuebingen.de/teaching/ss07/albi2/script/svm.pdf/view?searchterm>. Citado 2 vezes nas páginas 11 e 42.

ISERMANN, R. Process fault detection based on modeling and estimation methods—a survey. *automatica*, Elsevier, v. 20, n. 4, p. 387–404, 1984. Citado na página 25.

JAIN, A. K.; DUIN, R. P. W.; MAO, J. Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, Ieee, v. 22, n. 1, p. 4–37, 2000. Citado na página 35.

JAIN, R. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. [S.l.]: John Wiley & Sons, 2013. Citado na página 72.

JAMES, G. et al. *An introduction to statistical learning*. [S.l.]: Springer, 2013. v. 112. Citado na página 36.

JIA, F. et al. A neural network constructed by deep learning technique and its application to intelligent fault diagnosis of machines. *Neurocomputing*, Elsevier, v. 272, p. 619–628, 2018. Citado na página 23.

JIA, F. et al. Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data. *Mechanical Systems and Signal Processing*, Elsevier, v. 72, p. 303–315, 2016. Citado na página 54.

KELKAR, S.; KAMAL, R. Adaptive fault diagnosis algorithm for controller area network. *IEEE Transactions on Industrial Electronics*, IEEE, v. 61, n. 10, p. 5527–5537, 2014. Citado na página 26.

KHAN, S.; YAIRI, T. A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, Elsevier, v. 107, p. 241–265, 2018. Citado 2 vezes nas páginas 19 e 32.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Citado na página 49.

KONONENKO, I. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine*, Elsevier, v. 23, n. 1, p. 89–109, 2001. Citado na página 20.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1097–1105. Citado 2 vezes nas páginas 47 e 49.

KUMAR, A. et al. Classification of faults in web applications using machine learning. In: ACM. *Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*. [S.l.], 2017. p. 62–67. Citado 3 vezes nas páginas 11, 36 e 37.

KUNCHEVA, L. I. *Combining pattern classifiers: methods and algorithms*. [S.l.]: John Wiley & Sons, 2004. Citado na página 50.

LAN, G. An optimal method for stochastic composite optimization. *Mathematical Programming*, Springer, v. 133, n. 1-2, p. 365–397, 2012. Citado na página 49.

LANA, L. D. et al. Avaliação dos riscos do trabalho em altura na construção civil. *Revista Produção Online*, v. 14, n. 1, p. 344–363, 2014. Citado 2 vezes nas páginas 11 e 30.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Research, v. 521, n. 7553, p. 436–444, 2015. Citado na página 47.

LECUN, Y.; BENGIO, Y. et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, v. 3361, n. 10, p. 1995, 1995. Citado na página 47.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, IEEE, v. 86, n. 11, p. 2278–2324, 1998. Citado 2 vezes nas páginas 47 e 54.

LEI, J.; LIU, C.; JIANG, D. Fault diagnosis of wind turbine based on long short-term memory networks. *Renewable Energy*, Elsevier, v. 133, p. 422–432, 2019. Citado na página 53.

LEI, Y.; YUAN, Y.; ZHAO, J. Model-based detection and monitoring of the intermittent connections for can networks. *IEEE Transactions on Industrial Electronics*, IEEE, v. 61, n. 6, p. 2912–2921, 2014. Citado na página 26.

LEVY, R.; AROGETI, S. A.; WANG, D. An integrated approach to mode tracking and diagnosis of hybrid systems. *IEEE Transactions on Industrial Electronics*, IEEE, v. 61, n. 4, p. 2024–2040, 2014. Citado na página 26.

LI, S.; CAO, H.; YANG, Y. Data-driven simultaneous fault diagnosis for solid oxide fuel cell system using multi-label pattern identification. *Journal of Power Sources*, Elsevier, v. 378, p. 646–659, 2018. Citado na página 54.

LIBBRECHT, M. W.; NOBLE, W. S. Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, Nature Publishing Group, v. 16, n. 6, p. 321, 2015. Citado na página 20.

LIN, S.-W. et al. Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert systems with applications*, Elsevier, v. 35, n. 4, p. 1817–1824, 2008. Citado na página 43.

LIU, H.; MOTODA, H. *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Paperback. Springer, 2012. ISBN 146137622X,9781461376224. Disponível em: <http://gen.lib.rus.ec/book/index-.php?md5=50F1DA532B86F6AD77738FCB0973482E>. Citado na página 20.

LIU, Y. et al. A fta-based method for risk decision-making in emergency response. *Computers & Operations Research*, Elsevier, v. 42, p. 49–57, 2014. Citado na página 29.

LIU, Y.-K. et al. A fault diagnosis method based on signed directed graph and matrix for nuclear power plants. *Nuclear Engineering and Design*, Elsevier, v. 297, p. 166–174, 2016. Citado 2 vezes nas páginas 20 e 30.

LIU, Z. et al. Multi-fault classification based on wavelet svm with pso algorithm to analyze vibration signals from rolling element bearings. *Neurocomputing*, Elsevier, v. 99, p. 399–410, 2013. Citado 2 vezes nas páginas 33 e 53.

LIVANI, H.; EVRENOSOĞLU, C. Y. A fault classification method in power systems using dwt and svm classifier. In: IEEE. *Transmission and Distribution Conference and Exposition (T&D), 2012 IEEE PES*. [S.l.], 2012. p. 1–5. Citado na página 53.

LORENA, A. C.; CARVALHO, A. C. de. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, v. 14, n. 2, p. 43–67, 2007. Citado 2 vezes nas páginas 41 e 43.

MA, Y.; GUO, L.; CUKIC, B. A statistical framework for the prediction of fault-proneness. *Advances in machine learning application in software engineering*, p. 237–265, 2006. Citado na página 40.

MALHI, A.; GAO, R. X. Pca-based feature selection scheme for machine defect classification. *IEEE Transactions on Instrumentation and Measurement*, IEEE, v. 53, n. 6, p. 1517–1525, 2004. Citado na página 19.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 45.

MENDES, C. C. T. *Navigability estimation for autonomous vehicles using machine learning*. Tese (Doutorado) — Universidade de São Paulo, 2017. Citado 2 vezes nas páginas 11 e 48.

MICHIE, D. et al. (Ed.). *Machine Learning, Neural and Statistical Classification*. Upper Saddle River, NJ, USA: Ellis Horwood, 1994. ISBN 0-13-106360-X. Citado na página 50.

MITCHELL, T. M. *Machine Learning*. 1. ed. [S.l.]: McGraw-Hill, 1997. (McGraw-Hill series in computer science). Citado 3 vezes nas páginas 35, 44 e 49.

MORAIS, E. C. *Reconhecimento de padrões e redes neurais artificiais em predição de estruturas secundárias de proteínas*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2010. Citado na página 35.

MOSTAFA, S. A. et al. Implementing an expert diagnostic assistance system for car failure and malfunction. *IJCSI International Journal of Computer Science Issues*, v. 9, n. 2, p. 1694–0814, 2012. Citado na página 30.

NAMDARI, M.; JAZAYERI-RAD, H.; HASHEMI, S.-J. Process fault diagnosis using support vector machines with a genetic algorithm based parameter tuning. *Journal of Automation and Control*, v. 2, n. 1, p. 1–7, 2014. Citado na página 31.

NAN, C.; KHAN, F.; IQBAL, M. T. Real-time fault diagnosis using knowledge-based expert system. *process safety and environmental protection*, Elsevier, v. 86, n. 1, p. 55–71, 2008. Citado na página 32.

NASRABADI, N. M. Pattern recognition and machine learning. *Journal of electronic imaging*, International Society for Optics and Photonics, v. 16, n. 4, p. 049901, 2007. Citado na página 20.

OLSON, D. L.; DELEN, D. *Advanced data mining techniques*. [S.l.]: Springer Science & Business Media, 2008. Citado na página 37.

OLSSON, E.; FUNK, P.; XIONG, N. Fault diagnosis in industry using sensor readings and case-based reasoning. *Journal of Intelligent & Fuzzy Systems*, IOS Press, v. 15, n. 1, p. 41–46, 2004. Citado 2 vezes nas páginas 11 e 27.

OSHIRO, T. M. *Uma abordagem para a construção de uma única árvore a partir de uma Random Forest para classificação de bases de expressão gênica*. Tese (Doutorado) — Universidade de São Paulo, 2013. Citado na página 39.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado na página 66.

PEREZ, P. S. *Uma abordagem para a indução de árvores de decisão voltada para dados de expressão gênica*. Tese (Doutorado) — Universidade de São Paulo, 2012. Citado 2 vezes nas páginas 38 e 39.

POLYAK, B. T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, No longer published by Elsevier, v. 4, n. 5, p. 1–17, 1964. Citado na página 49.

POURBASHEER, E. et al. Application of genetic algorithm-support vector machine (ga-svm) for prediction of bk-channels activity. *European journal of medicinal chemistry*, Elsevier, v. 44, n. 12, p. 5023–5028, 2009. Citado na página 43.

PRATI, R. C.; BATISTA, G. E.; SILVA, D. F. Class imbalance revisited: a new experimental setup to assess the performance of treatment methods. *Knowledge and Information Systems*, Springer, v. 45, n. 1, p. 247–270, 2015. Citado na página 52.

QUINLAN, J. R. Induction of decision trees. *Machine learning*, Springer, v. 1, n. 1, p. 81–106, 1986. Citado na página 38.

RODRIGUES, J. F. *Seleção de características e aprendizado ativo para classificação de imagens de sensoriamento remoto*. Tese (Doutorado) — Universidade de São Paulo, 2015. Citado 2 vezes nas páginas 52 e 53.

ROKACH, O. M. L. *Data Mining With Decision Trees: Theory and Applications*. 2. ed. [S.l.]: World Scientific Publishing Company, 2015. Citado na página 39.

RUDER, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. Citado na página 82.

RUIJTERS, E.; STOELINGA, M. Fault tree analysis: a survey of the state-of-the-art in modeling, analysis and tools. *Computer science review*, Elsevier, v. 15, p. 29–62, 2015. Citado na página 29.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Neurocomputing: Foundations of research. *JA Anderson and E. Rosenfeld, Eds*, p. 696–699, 1988. Citado na página 46.

RUSSELL, E. L.; CHIANG, L. H.; BRAATZ, R. D. *Data-driven methods for fault detection and diagnosis in chemical processes*. [S.l.]: Springer Science & Business Media, 2012. Citado na página 28.

SAMPATH, M. et al. Diagnosability of discrete-event systems. *IEEE Transactions on automatic control*, IEEE, v. 40, n. 9, p. 1555–1575, 1995. Citado na página 26.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural networks*, Elsevier, v. 61, p. 85–117, 2015. Citado na página 47.

SCHRICK, D. van. Remarks on terminology in the field of supervision, fault detection and diagnosis. *IFAC Proceedings Volumes*, Elsevier, v. 30, n. 18, p. 959–964, 1997. Citado na página 23.

SESHADRINATH, J.; SINGH, B.; PANIGRAHI, B. K. Vibration analysis based interturn fault diagnosis in induction machines. *IEEE Transactions on Industrial Informatics*, IEEE, v. 10, n. 1, p. 340–350, 2014. Citado na página 33.

SHULBY, C. D. *RAMBLE: robust acoustic modeling for Brazilian learners of English*. Tese (Doutorado) — Universidade de São Paulo, 2018. Citado na página 50.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Citado 2 vezes nas páginas 57 e 82.

SMITH, D. S. et al. Robustness of quantitative compressive sensing mri: the effect of random undersampling patterns on derived parameters for dce-and dsc-mri. *IEEE transactions on medical imaging*, IEEE, v. 31, n. 2, p. 504–511, 2012. Citado na página 52.

SOTTILE, J.; HOLLOWAY, L. E. An overview of fault monitoring and diagnosis in mining equipment. *IEEE Transactions on Industry Applications*, IEEE, v. 30, n. 5, p. 1326–1332, 1994. Citado na página 23.

STAROSWIECKI, M. Model based fdi: the control approach. *Intelligence*, v. 12, n. 7, p. 6, 2001. Citado 2 vezes nas páginas 23 e 24.

STIBOREK, J.; PEVNỲ, T.; REHÁK, M. Multiple instance learning for malware classification. *arXiv preprint arXiv:1705.02268*, 2017. Citado na página 40.

STOJANOVIC, V. et al. Application of cuckoo search algorithm to constrained control problem of a parallel robot platform. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 87, n. 9-12, p. 2497–2507, 2016. Citado na página 19.

TAN, P.-N. et al. *Introduction to data mining*. [S.l.]: Pearson, 2018. Citado 3 vezes nas páginas 48, 51 e 72.

TIELEMAN, T.; HINTON, G. *Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning*. [S.l.], 2017. Citado na página 49.

UTGOFF, P. E.; STRACUZZI, D. J. Many-layered learning. *Neural Computation*, MIT Press, v. 14, n. 10, p. 2497–2529, 2002. Citado na página 46.

VALTIERRA-RODRIGUEZ, M. et al. Detection and classification of single and combined power quality disturbances using neural networks. *IEEE Transactions on Industrial Electronics*, IEEE, v. 61, n. 5, p. 2473–2482, 2014. Citado na página 31.

VAPNIK, V. *The nature of statistical learning theory*. [S.l.]: Springer science & business media, 2013. Citado na página 41.

VAPNIK, V. N.; VAPNIK, V. *Statistical learning theory*. [S.l.]: Wiley New York, 1998. v. 1. Citado 2 vezes nas páginas 41 e 42.

VENKATASUBRAMANIAN, V. et al. A review of process fault detection and diagnosis: Part iii: Process history based methods. *Computers & chemical engineering*, Elsevier, v. 27, n. 3, p. 327–346, 2003. Citado na página 24.

VITALE, R.; NOORD, O.; FERRER, A. A kernel-based approach for fault diagnosis in batch processes. *Journal of Chemometrics*, Wiley Online Library, v. 28, n. 8, 2014. Citado na página 31.

WANG, X. et al. Nonlinear pca with the local approach for diesel engine fault detection and diagnosis. *IEEE Transactions on Control Systems Technology*, IEEE, v. 16, n. 1, p. 122–129, 2008. Citado na página 31.

WANG, Y. et al. An intelligent approach for engine fault diagnosis based on hilbert–huang transform and support vector machine. *Applied acoustics*, Elsevier, v. 75, p. 1–9, 2014. Citado na página 54.

WEN, L. et al. A new convolutional neural network-based data-driven fault diagnosis method. *IEEE Transactions on Industrial Electronics*, IEEE, v. 65, n. 7, p. 5990–5998, 2018. Citado 2 vezes nas páginas 54 e 64.

WIATOWSKI, T.; BÖLCSKEI, H. A mathematical theory of deep convolutional neural networks for feature extraction. *IEEE Transactions on Information Theory*, IEEE, v. 64, n. 3, p. 1845–1866, 2018. Citado na página 50.

WILLSKY, A. S. A survey of design methods for failure detection in dynamic systems. *Automatica*, Elsevier, v. 12, n. 6, p. 601–611, 1976. Citado 2 vezes nas páginas 23 e 24.

WITTEN, I. H. et al. *Data Mining: Practical machine learning tools and techniques*. [S.l.]: Morgan Kaufmann, 2016. Citado na página 33.

YIN, S. et al. A review on basic data-driven approaches for industrial process monitoring. *IEEE Transactions on Industrial Electronics*, IEEE, v. 61, n. 11, p. 6418–6428, 2014. Citado na página 24.

YIN, S. et al. Data-based techniques focused on modern industry: An overview. *IEEE Transactions on Industrial Electronics*, IEEE, v. 62, n. 1, p. 657–667, 2015. Citado na página 31.

ZHANG, R. et al. Deep and shallow model for insurance churn prediction service. In: IEEE. *Services Computing (SCC), 2017 IEEE International Conference on*. [S.l.], 2017. p. 346–353. Citado na página 54.

ZHAO, R. et al. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, Elsevier, v. 115, p. 213–237, 2019. Citado na página 19.

ZHAO, Y.; ZHANG, Y. Comparison of decision tree methods for finding active objects. *Advances in Space Research*, Elsevier, v. 41, n. 12, p. 1955–1959, 2008. Citado na página 40.

ZHU, X.; GOLDBERG, A. B. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, Morgan & Claypool Publishers, v. 3, n. 1, p. 1–130, 2009. Citado na página 36.