Machine Learning Applied to the Classification of Technical Inspection Recommendations Regarding the Trend to Increase Criticality

André, L. A., Amaral*
PUC-Rio
andre.luis.azevedo.amaral@gmail.com

André, L. R., de Araújo
PUC-Rio
andre.r.araujo@gmail.com

Bruno Tomazela
PUC-Rio
btomazela@gmail.com

Tatiana Escovedo
PUC-Rio
tatiana@inf.puc-rio.br

Marcos Kalinowski
PUC-Rio
kalinowski@inf.puc-rio.br

ABSTRACT

Context: The problem of prioritizing maintenance activities has become a subject of great interest to the industry in a highly competitive scenario where profitability, productivity, and safety are sought in highly automated plant operations. **Problem:** The operational unit object of this study faces a high demand for maintenance services, indicated by the technical inspection recommendations (TIR) and the risks arising from not meeting them in a timely manner, with the consequent operational and financial losses. **Solution:** This article proposes a tool to support the prioritization of equipment maintenance activities through a machine learning model that classifies TIRs according to their criticality, thus providing a second opinion for the classification made by the inspector and, therefore, indicating those that have a greater tendency to escalate in criticality. **IS Theory:** This work is associated with the Theory of the knowledge-based company, assisting in decision-making and efficiency in the application of resources. **Method:** To construct the Machine Learning model, algorithms and ensembles used in classification problems and natural language processing methods were used, considering that one of the most relevant attributes of the dataset is a free textual field. **Summary of Results:** After carrying out several experiments, we arrived at the best performance model, with a recall of 87.08% and an accuracy of 95.19%, values understood as promising for the implementation of the tool in production. **Contributions and Impact in the IS area:** The main contribution was to identify the feasibility of using machine learning models for the classification of technical inspection recommendations regarding the trend to increase criticality.

CCS CONCEPTS

• **Information systems**; • **Information systems applications**;

*Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SBSI '23, May 29–June 01, 2023, Maceió, Brazil
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0759-9/23/05…$15.00
https://doi.org/10.1145/3592813.3592884

KEYWORDS

Industrial maintenance, Service prioritization, Machine learning, Classification models, Natural Language Processing

ACM Reference Format:
André, L. A., Amaral, André, L. R., de Araújo, Bruno Tomazela, Tatiana Escovedo, and Marcos Kalinowski. 2023. Machine Learning Applied to the Classification of Technical Inspection Recommendations Regarding the Trend to Increase Criticality. In *XIX Brazilian Symposium on Information Systems (SBSI '23), May 29–June 01, 2023, Maceió, Brazil.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3592813.3592884

1 INTRODUÇÃO

Dentre as atividades relacionadas à manutenção dos equipamentos de plantas industriais está a de inspeção de equipamentos, que permite verificar regularmente a integridade estrutural de um equipamento, visando assegurar a condição segura de operação e, eventualmente, gerando recomendações de inspeção que antecipem medidas de intervenção antes de uma falha do equipamento [1]. Este trabalho se concentra em uma unidade de negócio (UN) de uma grande empresa de óleo e gás (O&G), que possui 10 plantas industriais (no caso, unidades estacionárias de produção – UEPs – de O&G) sob sua gestão e, portanto, cuja manutenção está sob sua responsabilidade.

A atividade de inspeção gera Recomendações Técnicas de Inspeção (RTIs), que são categorizadas quanto a sua criticidade (probabilidade e impacto da falha) como A, B, C e D, sendo A a categoria de maior criticidade e D a de menor. A categoria A requer uma manutenção imediata, pois pode implicar na parada da UEP, consequentemente, perdas de produção e, em última análise, perdas financeiras de alta monta. A categoria B tem a segunda maior prioridade de manutenção, requer prazos de atendimento curtos e, se necessário, monitoramentos frequentes. As categorias C e D têm menor prioridade, prazos longos para atendimento e monitoramentos menos frequentes.

Ocorre que, em um universo de cerca de uma dezena de milhares de RTIs, enquanto as RTIs de categoria A estão na ordem de unidades e as de categoria B de poucas centenas, as de categorias C e D estão na ordem de alguns milhares e, frequentemente, têm sua criticidade escalada para B ou mesmo A, em função da deterioração da condição do equipamento, antes da manutenção recomendada ocorrer ou mesmo de haver um novo monitoramento. Como exemplos desta deterioração da condição, pode-se citar os

casos decorrentes do agravamento da corrosão de equipamentos e instalações, como os de "suportação", onde o processo de corrosão pode evoluir de um estágio leve para um estágio que pode implicar na perda de suporte de uma tubulação e acidentes graves, com a sua queda. Ou ainda casos de corrosão em tubulações, onde, devido ao aumento da corrosão, uma linha pode perder sua função no processo, sendo necessária a interrupção da vazão do fluido para o seu reparo. Desta forma, o problema apresentado é de priorização das RTIs de categorias C e D.

Atualmente, este processo de priorização não é feito de forma sistemática, devido ao grande volume de RTIs, observando-se aumento da criticidade já em um momento em que seja exigida uma ação imediata, eventualmente impactando a continuidade operacional do equipamento e, consequentemente, até mesmo exigindo paradas de produção e resultando em prejuízos financeiros. Além disso, com o aumento da criticidade, a complexidade da manutenção e seu custo tendem a subir.

Este trabalho propõe uma solução de *Machine Learning – ML* (aprendizado de máquina) para a priorização das RTIs de categorias C e D, identificando aquelas com maior tendência de escalarem para as categorias A e B, a partir dos dados de RTIs anteriores, tanto que tenham escalado, quanto que não tenham escalado ao longo do tempo.

É explorado o problema de classificação das RTIs quanto à sua tendência de aumento de criticidade e técnicas de *Natural Language Processing - NLP* (Processamento de Linguagem Natural), visto que campos importantes na caracterização de uma RTI são textuais livres.

Espera-se, com este trabalho, validar a aplicação destas técnicas no problema de priorização da manutenção de equipamentos e contribuir com a unidade na minimização de riscos operacionais e perdas financeiras decorrentes de suas falhas.

Este artigo está organizado em 4 seções adicionais. Na Seção 2, são discutidos trabalhos correlatos. Na Seção 3, é apresentada a solução proposta para o problema e na Seção 4 são apresentados os experimentos realizados para a implementação da solução. Por fim, na Seção 5, são tecidas as conclusões, considerações finais e propostas de trabalhos futuros.

2 REVISÃO DA LITERATURA

Recentemente, com o surgimento do conceito de Indústria 4.0, caracterizada pela integração de sistemas físicos e digitais através da automação de plantas industriais, profusão de sensores, estabelecimento de conexões através de plataformas *IOT* (*Internet of Things* – Internet da Coisas), associado ao desenvolvimento do campo de *Machine Learning*, a manutenção preditiva (*PdM – Preditive Maintenance*) tem se tornado uma tendência, devido às suas vantagens sobre a manutenção corretiva ou preventiva, tais como a detecção de falhas em um estágio inicial, sem paradas dos equipamentos, permitindo uma otimização de seu uso e gerenciamento [2].

O presente trabalho, por outro lado, não se baseia em dados de automação de planta, mas em registros históricos de RTIs, que contém informações principalmente textuais, preenchidas por profissionais de manutenção, como informações sobre o equipamento inspecionado, problemas identificados e recomendações.

Nesta temática, LUCHETTI [3] teve como objetivo elaborar, por meio de dados de manutenção de máquinas, um modelo de ML para predizer se a manutenção de uma máquina será realizada dentro do prazo desejado, sendo uma abordagem diferente de priorização, não procurando identificar o que é crítico, mas o que não vai ter o prazo de manutenção cumprido.

O autor observa que o modelo proposto deve ser usado como uma ferramenta de auxílio para tomada de decisões relacionadas ao trabalho de manutenção na empresa, por meio da interpretação dos resultados produzidos, como se fosse uma "segunda opinião", não devendo-se descartar uma análise qualitativa de especialistas, visto que o algoritmo não considera todos os fatores existentes relacionados a uma manutenção ou a outras prioridades da própria empresa, além de não acertar em todas as classificações. Para a escolha final do modelo, foi priorizado o modelo com maior precisão, visto que sua classe "positiva" é "Dentro do Prazo" e, sendo assim, os falsos positivos são menos toleráveis do que os falso negativos. Esta situação é oposta à apresentada no presente trabalho, onde a classe positiva é "Crítica" e os falsos negativos são menos toleráveis do que os falsos positivos, trazendo uma importância maior para a métrica *recall*.

Assim como o campo de manutenção de equipamentos, outras atividades humanas demandam priorização de serviços que, de forma análoga, também é uma necessidade decorrente da alta demanda e recursos limitados. Por exemplo, na área de saúde, há trabalhos de criação de modelos de *Machine Learning* que visam predizer a necessidade de internação de pacientes, fazendo relação entre internações e a urgência definida em processo de triagem [4], assim como predizer as cirurgias eletivas, em uma lista de espera, que se tornarão cirurgias urgentes, caso não sejam priorizadas a tempo [5].

No presente trabalho, a natureza do *target* (atributo-alvo) é diferente daquela apresentada em [4], considerando que aqui pretende-se predizer quais RTIs escalarão em criticidade, enquanto na predição de internações, este conceito de criticidade não está claramente presente, mostrando maior correlação com as informações médicas coletadas dos pacientes e menor correlação com a classificação de urgência do processo de triagem. Observa-se o uso de *undersampling* para o balanceamento do *dataset*, ao contrário do presente trabalho, onde é usado um *oversampling*, esta diferença se justifica pela relação entre número de atributos, 17, e de registros, ~177.000 (1:10.000). Aqui essa relação é bem menor, com 14 atributos e ~21.000 registros (1:1.500). Isto poderia ser diferente com um *dataset* contendo dados de outras UNs além da que é objeto deste estudo, o que poderá ser abordado em trabalhos futuros.

Já o trabalho de MORAES [5] possui forte correlação em objetivos com o trabalho aqui desenvolvido, pois procura identificar pacientes que necessitarão de cirurgia urgente em uma fila de espera de pacientes para cirurgia eletiva. Feita a ressalva que nesse caso está se lidando diretamente com vidas humanas, a analogia existente é que o presente trabalho procura identificar equipamentos que necessitarão de manutenção prioritária (RTIs de categorias A e B) em uma fila de espera de equipamentos para manutenção não prioritária (RTIs de categorias C e D).

O autor cita diversas características do problema, assim como pontos de atenção tratados para a construção do modelo, similares aos presentes neste trabalho, tais como:

- problema de classificação binária;
- uma classe "positiva" (urgente/crítica) e uma "negativa" (eletiva/não crítica);
- necessidade de balanceamento das classes (usa o undersampling através da eliminação aleatória de amostras da classe majoritária, enquanto o presente trabalho usa oversampling com geração de amostras sintéticas da classe minoritária);
- a importância das métricas de recall (sensibilidade) e precisão, considerando a presença de uma classe "positiva" e o maior impacto da predição de muitos falsos negativos do que de muitos falsos positivos;
- uso de ensembles e algoritmos baseados em árvores.

3 SOLUÇÃO PROPOSTA

A metodologia usada neste trabalho se baseia nas etapas do esquema básico de um projeto de Ciência de Dados, apresentado em [6, 7], restringindo o seu escopo às primeiras quatro etapas, onde serão apresentados os resultados dos experimentos de modelagem e inferência. As etapas de Pós-processamento em diante serão realizadas em um trabalho futuro junto à área de negócio. Para a realização de cada etapa, vários métodos foram utilizados, como será demonstrado a seguir.

3.1 Problema, Necessidade ou Ideia

Conforme mencionado anteriormente, o problema de negócio tratado por este trabalho é essencialmente um problema de priorização de serviços a serem executados. Devido ao grande volume de RTIs de categoria C e D e de sua relativa baixa criticidade no momento do registro, o seu monitoramento não tem a frequência existente para as RTIs de categorias A e B.

Desta forma, a identificação de um aumento de sua criticidade pode ocorrer em um momento em que já seja exigida uma ação imediata, ou mesmo impactar na continuidade operacional do equipamento e, consequentemente, até mesmo paradas de produção e prejuízos financeiros. Além disso, com o aumento da criticidade, a complexidade da manutenção e seu custo tendem a subir.

O objetivo do trabalho então é identificar, dentre as RTIs de categorias C e D (aqui classificadas como "não críticas"), aquelas com maior tendência de escalarem para as categorias A e B (aqui classificadas como "críticas"), auxiliando os responsáveis pela manutenção na priorização das intervenções que demandem maior atenção. Tratamos este problema como sendo de Classificação e construímos modelos baseados em algoritmos de aprendizado supervisionado, onde o *dataset* apresenta as características das RTIs concluídas.

Inicialmente, este trabalho propõe a definição de um *target* com a informação se a RTI é "crítica" ou "não crítica" e, desta forma, o modelo pretende prover uma "segunda opinião" para os responsáveis pela manutenção, ou seja, classificar uma RTI de categorias C ou D (portanto, originalmente, "não crítica") como "crítica" ou "não-crítica". Em um trabalho futuro, pretende-se avaliar o comportamento das RTIs no tempo, observando a evolução de sua criticidade e, neste caso, trabalhar diretamente com um *target* "escalou" ou "não escalou".

Adicionalmente, considerando que talvez a mais relevante das características de uma RTI é um campo textual livre, onde o profissional responsável pela inspeção registra uma série de informações relacionadas à recomendação feita, fazemos uso de técnicas de NLP, buscando gerar novas características que serão testadas e incorporadas ao *dataset* estruturado.

A solução proposta, então, faz uso das técnicas mencionadas para predizer quais as RTIs que escalarão de criticidade, permitindo a priorização das intervenções pela equipe de manutenção, otimização dos esforços na atividade, redução de riscos operacionais e de custos, assim como mitigação de riscos de produção e de segurança, meio-ambiente e saúde.

3.2 Coleta e Análise de Dados

O *dataset* usado neste projeto foi gerado com base em consulta feita em cima de um modelo SAP BW (base informacional) de PM (módulo Planejamento da Manutenção), que por sua vez é alimentado por dados do módulo de PM do SAP ECC (base transacional). Vale ressaltar que os dados passaram por pequenos ajustes feitos pela área de negócio, mas que como foram poucos, espera-se que não tenham nenhum impacto no trabalho. Os dados, já ajustados, foram fornecidos em uma planilha Excel.

Foi verificado que o *dataset* fornecido possuía 37.332 instâncias e 36 atributos (ver Tabela 1), em sua grande maioria, categóricos, sendo alguns do tipo data, e apenas dois atributos numéricos, verificando-se ainda que muitos atributos apresentavam valores faltantes (*missing values*).

Com o apoio de um especialista do negócio, estatísticas descritivas e visualizações unimodais e multimodais, os atributos foram entendidos conceitualmente, avaliadas as suas distribuições de valores, correlações e foram identificados aqueles mais relevantes para o problema de negócio.

Por fim, verificou-se que a variável *target* era muito desbalanceada e que as classes mais relevantes eram as minoritárias. Relembrando que as classes A (465 registros) e B (2518 registros) são consideradas "Críticas" e as C (16.046 registros) e D (18.303 registros) "Não críticas". Esse desbalanceamento já sinalizou a necessidade de se usar algum método de balanceamento e de considerar o uso de outras métricas além da acurácia, como *recall* ou precisão.

3.3 Pré-processamento

Com os dados analisados e entendidos, nesta etapa foi realizado o seu pré-processamento, visando produzir uma nova versão do *dataset* onde os dados estejam selecionados conforme a sua relevância para o modelo, limpos e enriquecidos através de transformações.

Inicialmente, foi realizada a atividade de feature selection (seleção de atributos), onde, com base nas informações levantadas junto ao especialista do negócio e ao estudo de correlação entre os atributos, foi selecionado um subconjunto de características dos dados originais do dataset para serem mantidos no modelo.

Para o estudo de correlação entre os atributos, foram usadas as bibliotecas Dython e Featurewiz, que trabalham tanto com dados numéricos, quanto com dados categóricos, algo percebido como raro nas referências pesquisadas, onde a maioria das bibliotecas trabalhavam apenas com atributos numéricos. Este estudo demonstrou a alta correlação entre alguns atributos relacionados aos equipamentos, tendo-se descartado dois deles, "Área Operacional" e "TAG".

Table 1: Atributos do *Dataset* fornecido

Atributo	Descrição	Atributo	Descrição
Área	Constante "SPIE"" (Existem várias área na UN e o dataset está limitado com as RTIs sob responsabilidade do SPIE)	Objeto	Código de um equipamento ou local de instalação
Área operacional	Campo usado pela operação para localizar os equipamentos na plataforma	Planejamento MONI	Data limite para nova reinspeção (caso a recomendação não seja executada)
Classe	Classificador geral dos equipamentos em válvulas, tubulação, vasos de pressão, tanques, etc.	Plataforma	Unidade de produção que a recomendação foi emitida
Criticidade	Categorização de quão critica é a inspeção	Possui reparo?	indica se o equipamento possui reparo
Data CTEC	Data em que o inspetor concluiu a inspeção e emitiu a RT	Recurso para Monitoramento	Recuros necessários para o monitoramento
Data Fim Monitoramento	Data limite em que a reinspeção "atesta" o laudo dado no "monitoramento"	REPARO.CODIGO	Código sequencial da ação no SAP, usada para identificar a linha do registro
Data MEDE	Data em que a equipe de manutenção executa a recomendação	REPARO.DENOMINACAO	Texto livre (máx 20 caracteres) escrito pelo registrador do reparo.
Data PH	Data em que o PH aprovou a RTI	REPARO.DT_CRIACAO	Data de registro de reparo temporário
Denominação LI	Denominação do local de instalação	REPARO.DT_FIM	Data em que acaba a certificação do reparo temporário aplicado naquela recomendação
Equip Status Sis	Status de fluxo do ERP SAP. Usada para identificar se o equipamento está operando ou com algum código de manutenção	Responsável Execução	Campo de códigos usado pela pessoa que executa, cada INSUP(área) possui um padrão
Equip Status Usu	Status de fluxo do ERP SAP. Usada para identificar se o equipamento está operando ou com algum código de manutenção	RTI	Código da Recomendação Técnica de Inspeção
Grupo Codes	Código usado pela equipe de planejamento de paradas de manutenção	Status Geral	Status macro do processo de acompanhamento das recomendações
Impacto de Falha	Impacto da falha onde 6 significa maior impacto e 1 menor	Status Sistema	Status de fluxo do ERP SAP, usado para identificar em qual etapa do fluxo da recomendação se encontra
Local de Instalação	Código SAP que define uma posição na árvore de locais	Status Usuário	Status de fluxo do ERP SAP, usado para identificar em qual etapa do fluxo da recomendação se encontra
Medida	Sequencial dos itens de inspeção	TAG	Código do equipamento / objeto a ser inspecionado
MONI.IMPOSSIBILIDADE	Campo usado para informar quando há alguma impossibilidade de realizar o monitoramento	Texto breve	Resumo da recomendação técnica derivada da inspeção realizada pelo engenheiro
Monitorabilidade	Indica se equipamento é monitorável	TEXTO_LONGO	Recomendação técnica derivada da inspeção realizada pelo engenheiro
NOTA.AUTOR	Campo livre que é preenchido pelo inspetor ou PH na abertura da recomendação	Vencimento	Vencimento da RTI. Por padrão elas são emitidas com um prazo para adequação A=0,B=90,C=360,D=720 dias

Table 2: Atributos do *Dataset* após o pré-processamento

Atributo	Descrição	Atributo	Descrição
Classe	Classificador geral dos equipamentos em válvulas, tubulação, vasos de pressão, tanques, etc.	Plataforma	Unidade de produção que a recomendação foi emitida
Criticidade	Categorização de quão critica é a inspeção	Possui reparo?	indica se o equipamento possui reparo
Grupo Codes	Código usado pela equipe de planejamento de paradas de manutenção	Responsável Execução	Campo de códigos usado pela pessoa que executa, cada INSUP(área) possui um padrão
Idade	Indica a idade da RTI em dias (atributo calculado com base em outros atributos)	Sistema	Parte do atributo local de instalação que era muito específico (atributo calculado com base em outros atributos)
Impacto de Falha	Impacto da falha onde 6 significa maior impacto e 1 menor	TEXTO_LONGO	Recomendação técnica derivada da inspeção realizada pelo engenheiro
NOTA.AUTOR	Campo livre que é preenchido pelo inspetor ou PH na abertura da recomendação	Vencida	Indica se a RTI está vencida ou não. (atributo calculado com base em outros atributos)
Objeto	Código de um equipamento ou local de instalação	Crítico	Target - 1 para Crítico e 0 para não crítico (Calculado com base no atributo Criticidade)

Na atividade de feature engineering (engenharia de atributos), alguns dados foram transformados visando sua adequação ao modelo ou a sua substituição por dados derivados deles, porém, com maior relevância para o modelo.

A classe do dataset foi transformada, substituindo-se os valores A e B por "Crítico" (1) e os valores C e D por "Não Crítico" (0), o que caracteriza o problema como sendo de classificação binária. Foram ainda criados os atributos "Sistema" (parte mais relevante do atributo hierárquico "Local de Instalação"), "Idade" e "Vencida" (informações entendidas como relevantes e derivadas dos diversos atributos do tipo data do dataset original).

O tratamento de missing values (valores faltantes) não foi necessário, pois, a esta altura, em função da exclusão de diversos atributos considerados irrelevantes para o modelo, apenas o atributo textual apresentava missing values, porém, este atributo será pré-processado com técnicas de NLP, como será apresentado mais adiante.

Foram identificadas 131 instâncias duplicadas, provavelmente, decorrentes das exclusões de atributos irrelevantes realizadas anteriormente. Estas instâncias foram eliminadas.

Nesta etapa, também foram selecionadas apenas as instâncias concluídas do dataset (atributo "Status Sistema" = 'MEDE MSUC') para uso no treinamento do modelo e foi realizado o holdout (separação do dataset em bases de treino e teste), separando-se 20% das instâncias para o dataset de teste.

Ao final desta etapa de pré-processamento, o dataset passou a contar com 14 atributos e 26.186 instâncias, sendo 20.948 para treino. Os atributos do dataset após o pré-processamento são apresentados na Tabela 2.

Antes de serem iniciados os experimentos, foi selecionada a técnica a ser utilizada para o balanceamento do *dataset* e as métricas de qualidade a serem utilizadas. Após os últimos pré-processamentos realizados, tínhamos um cenário de desbalanceamento de classes e, como modelos de classificação não têm bom desempenho com *datasets* desbalanceados, foram utilizadas técnicas de reamostragem de instâncias que permitem o seu balanceamento. No caso deste trabalho, em função de não se contar com um grande *dataset* em número de instâncias, optou-se pelo uso da técnica de *oversampling*, gerando-se novas instâncias sintéticas da classe minoritária (crítica) com a técnica *SMOTE*, desta forma, buscando-se evitar o *overfitting* causado pelo *oversampling*. Esta técnica será aplicada em todas as versões do *dataset* usadas nos experimentos.

Em relação às métricas de desempenho, para um problema de classificação binária, temos duas classes possíveis, onde uma delas pode ser considerada a "positiva" e a outra a "negativa". No contexto deste trabalho, a classe "crítica" foi considerada a classe "positiva". Considerando que um falso positivo implicará apenas na priorização de uma manutenção não crítica que de qualquer forma necessitaria ser realizada, enquanto um falso negativo poderá levar à não priorização de uma manutenção crítica, com as consequências negativas já abordadas, a métrica de *recall* ganha maior relevância, pois seu valor é tanto maior quanto menor for o número de falsos negativos.

Este trabalho, então, apura e avalia em seus experimentos diversas métricas de desempenho, como acurácia, precisão, *recall* e *F1-score*, porém, prioriza os modelos de maior *recall*.

Table 3: Análise de tokens por corpus

corpus	# tokens	# tokens diferentes
corpus_tokenized	5.200	0
corpus_lem	4.498	1.512
corpus_bigram	5.656	748
corpus_bigram_lem	5.048	2.193
corpus_trigram	5.749	890
corpus_trigram_lem	5.131	2.313

3.4 Pré-processamento do campo textual

Numa análise inicial do campo textual foram identificadas 92 instâncias com textos nulos. Essas instâncias foram removidas para geração do *corpus* submetido ao pré-processamento do campo textual. No final do pré-processamento do campo textual essas instâncias serão devidamente tratadas e reinseridas nos *datasets* que serão encaminhados para os experimentos de classificação das RTIs. A forma como as instâncias com texto nulo foram reinseridas no *dataset* será explicada ao longo desta subseção.

Antes de qualquer pré-processamento, o *corpus* possuía 83.681 tokens distintos, considerando cada *token* como uma palavra precedida e seguida por um ou mais caracteres de espaço em branco. O primeiro tratamento feito no campo texto foi remover: (i) acentuação, (ii) palavras com dois ou menos caracteres (por exemplo, "a" e "um"), (iii) pontuação, (iv) palavras que misturam números e caracteres alfabéticos (por exemplo, códigos como "T18"), (v) palavras que aparecem uma única vez no *corpus* e (vi) *stop words* – a lista de *stop words* inclui artigos, interjeições e verbos auxiliares como "ser", "estar" e "haver". Após essa etapa, foram identificados 5.200 *tokens* distintos compondo o *corpus* de treino. Toda a análise a seguir tem como base esse *corpus*, que será chamado de *corpus_tokenized*.

Em seguida, foi realizada a lematização do *corpus_tokenized*, cujo *corpus* resultante foi denominado *corpus_lem*. O último tratamento realizado foi a identificação de *bigrams* e *trigrams*; essa etapa foi feita tanto para o *corpus_tokenized* quanto para o *corpus_lem*, visando gerar maior diversidade de *corpus* a serem avaliados. O *corpus* gerado por meio da aplicação de *bigrams* no *corpus_tokenized* e no *corpus_lem* foram denominados, respectivamente, *corpus_bigram* e *corpus_bigram_lem;* da mesma forma, foram gerados os *corpus* *corpus_trigram* e *corpus_trigram_lem*.

A quantidade de *tokens* únicos e a quantidade de tokens diferentes em cada corpus está ilustrada na Tabela 3. Para cálculo da identificação de tokens diferentes foi utilizado como base o *corpus_tokenized*. Ou seja, o *corpus_lem* possui 1512 tokens diferentes do *corpus_tokenized*. Com essa análise, é possível verificar que os *corpus* gerados são significativamente diferentes entre si.

Na Tabela 4 está sintetizada uma análise da quantidade de *tokens* por documento em cada *corpus*. No *corpus_tokenized*, utilizado como linha de base para todo o pré-processamento do campo textual, existem em média 21 *tokens* por documento com desvio padrão 18; o menor documento nesse *corpus* possui 2 *tokens* e o maior documento possui 882 *tokens*. Pelo desvio padrão, podemos perceber (i) que o tamanho dos documentos em cada *corpus* é bastante variado e (ii) que o processo de lematização está gerando documentos vazios pois os *tokens* originais não pertencem à classe gramatical permitida no

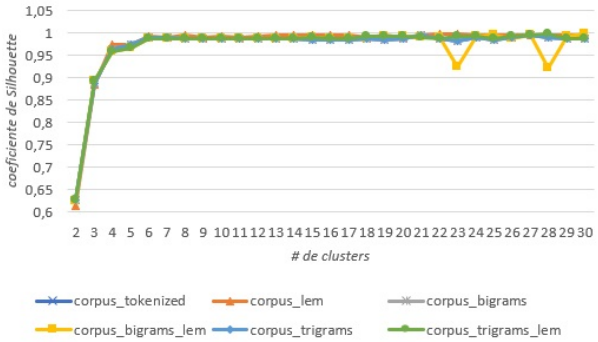

Figure 1: Coeficiente de *Silhouette* de acordo com a variação da quantidade de clusters em cada *corpus*

processo de lematização. Neste ponto vale ressaltar que, no processo de lematização, foram considerados apenas *substantivos, adjetivos, verbos* e *advérbios*; demais classes gramaticais foram eliminadas.

Com base nesses seis *corpus*, foram adotadas duas estratégias para transformar o campo textual em dado quantitativo. A primeira estratégia consistiu na clusterização dos documentos e a segunda na identificação dos tópicos presentes em cada documento.

Na primeira estratégia, foi utilizada a biblioteca *SKLearn*. A clusterização dos documentos foi feita utilizando o algoritmo *K-Means* com base na métrica TF-IDF. A qualidade dos *clusters* foi avaliada com base no coeficiente de *Silhouette*. Para cada *corpus*, foram utilizados os *n* tokens com maior TF-IDF para encontrar o *m* cluster. O hiperparâmetro *n* foi testado com 2, 5 e 10 *tokens* com maior TF-IDF e o número de clusters foi variado de 2 a 30. Para todos os *corpus*, o melhor resultado foi encontrado com *n* igual a 2; os dois *tokens* que contribuíram para a clusterização foram "isometrico" e "pintar". Dessa forma, na Figura 1 é ilustrado o coeficiente de *Silhouette* para a variação de *clusters*, considerando como *feature* para o agrupamento o TF-IDF dos tokens "isometrico" e "pintar".

Como pode ser observado na Figura 1, o resultado do coeficiente de *Silhouette* foi próximo de um a partir de cinco clusters. Os melhores resultados para cada *corpus* foram resumidos na Tabela 5.

Com base nos resultados apresentados, foi utilizado o *corpus_lem* para transformar o texto em dado quantitativo. Essa transformação foi feita calculando a distância de cada documento ao centroide de cada *cluster*. Dessa forma, utilizando a estratégia de clusterização, o campo textual foi substituído por 23 números que representam a posição espacial do documento no espaço calculado com base no TF-IDF dos dois *tokens* mais representativos no *corpus*.

Na segunda estratégia, a identificação dos tópicos foi feita utilizando o algoritmo LDA das bibliotecas *Gensim* e *Mallet*. Como essas bibliotecas utilizam como base o mesmo algoritmo, variando apenas o método de treinamento, iremos apresentar as análises dos resultados usando apenas a biblioteca *Gensim* pois foi a que obteve os melhores resultados na classificação, como será visto na Seção 4. O primeiro experimento realizado foi para identificar o *corpus* e o número de tópicos com o melhor resultado para a métrica de coerência utilizando o conjunto de parâmetros padrão do *Gensim*. Nesse experimento, o número de tópicos foi variado de 2 a 10, conforme resultado apresentado na Figura 2. O melhor resultado foi

Table 4: Análise de tokens por documento

corpus	média	desvio padrão	mínimo	máximo
corpus_tokenized	21	18	2	882
corpus_lem	18	16	0	802
corpus_bigram	20	17	2	855
corpus_bigram_lem	17	15	0	766
corpus_trigram	19	16	2	847
corpus_trigram_lem	17	14	0	760

Table 5: Melhores resultados da clusterização

corpus	# de clusters	Coeficiente de *Silhouette*
corpus_tokenized	21	0.995
corpus_lem	23	0.997
corpus_bigram	21	0.995
corpus_bigram_lem	30	0.997
corpus_trigram	21	0.995
corpus_trigram_lem	28	0.997

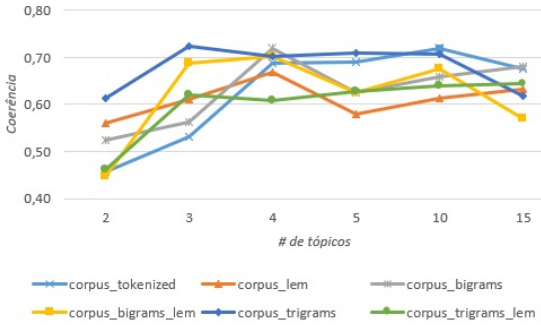

Figure 2: Coerência dos tópicos gerada pelo LDA

uma coerência igual a 72% obtido para *corpus_tokenized* com 10 tópicos. Como parte desse primeiro experimento, foram identificados *tokens* irrelevantes que estavam contribuindo para os tópicos modelados. Esses *tokens* foram incluídos na lista de *stop-words* visando melhorar a semântica dos tópicos.

Como o pré-processamento do *corpus_tokenized* realiza menos alterações no documento do que o pré-processamento do *corpus_trigrams*, o *corpus_tokenized* foi utilizado para transformar o campo textual em quantitativo.

Para transformar o campo textual em números, foi identificado o percentual de contribuição de cada tópico para documento. Dessa forma, cada documento foi transformado em 10 números, pois o melhor resultado foi obtido com 10 tópicos.

Os principais tokens utilizados na modelagem de cada um dos 10 tópicos foram:

- tópico_1 = {linha, deck, localizacao, main, modulo}
- tópico_2 = {aco, cloves, seidel, pressao, recomenda}
- tópico_3 = {linha, substituir, isometrico, trecho, nota}
- tópico_4 = {pintar, tratar, corrosao, isometrico, tubulacao}

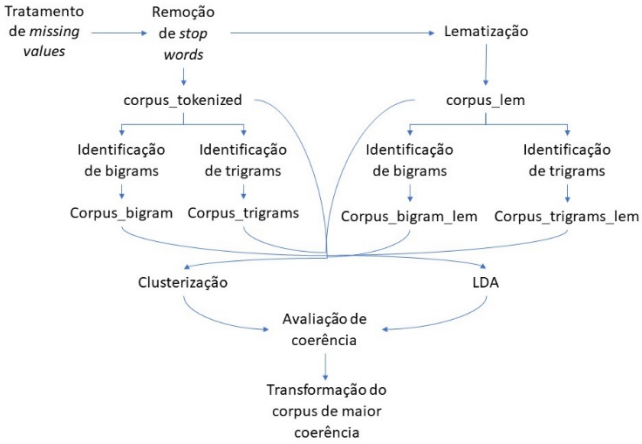

Figure 3: Fluxograma Pré-processamento NLP

- tópico_5 = {gilberto, antonio, adamatti, rti, status}
- tópico_6 = {valvula, psv, flange, dreno, conexao}
- tópico_7 = {suporte, instalar, suportes, regiao, acesso}
- tópico_8 = {pintura, tratamento, foto, linha, realizar}
- tópico_9 = {solda, asme, espessura, soldada, sela}
- tópico_{10} = {estojos, porcas, placa, elastomero, recomendacao}

Dessa forma, foram gerados dois conjuntos de treino diferentes com base na substituição do texto por valores numéricos gerados por meio da clusterização dos documentos e identificação de tópicos utilizando LDA. A Figura 3 mostra um fluxograma com as etapas do pré-processamento do campo textual.

Table 6: Resultados primeira fase de experimentos

Classificador	Pré-processamento	Hiperparâmetros	Acurácia
Bag_DecTree_Knn	StandardScaler_OHE	base_estimator: DecisionTreeClassifier, n_estimators: 10	95,60%
Bag_DecTree_Knn	MinMaxScaler_OHE	base_estimator: DecisionTreeClassifier, n_estimators: 10	95,52%
RanFor	StandardScaler_OHE	max_features: 10, n_estimators: 10	95,50%
RanFor	MinMaxScaler_OHE	max_features: 10, n_estimators: 10	95,41%
ExtTree	MinMaxScaler_OHE	max_features: 10, n_estimators: 10	94,95%
ExtTree	StandardScaler_OHE	max_features: 10, n_estimators: 10	94,86%
KNN	StandardScaler_OHE	metric: minkowski, n_neighbors: 5	93,11%
KNN	MinMaxScaler_OHE	metric: minkowski, n_neighbors: 5	92,67%
CART	MinMaxScaler_OHE	criterion: entropy, max_depth: 7	89,80%
CART	StandardScaler_OHE	criterion: entropy, max_depth: 7	89,79%
SVM	StandardScaler_OHE	C: 0.1, kernel: linear	86,73%
LR	StandardScaler_OHE	-	85,91%
LR	MinMaxScaler_OHE	-	85,36%
GradBoost	StandardScaler_OHE	n_estimators: 10	85,11%
GradBoost	MinMaxScaler_OHE	n_estimators: 10	85,11%
SVM	MinMaxScaler_OHE	C: 0.1, kernel: linear	84,39%
NB	StandardScaler_OHE	var_smoothing: 0.01	73,32%
NB	MinMaxScaler_OHE	var_smoothing: 0.01	73,24%

3.5 Considerações finais do pré-processamento

Neste ponto já tínhamos todo o *dataset* tratado e estávamos prontos para iniciar os experimentos em busca do melhor modelo de classificação de RTIs. Estas atividades correspondem à etapa de **Modelagem e Inferência** e serão apresentadas na seção seguinte.

4 APLICAÇÃO DA SOLUÇÃO

A construção do modelo foi iterativa, em diferentes fases, conforme eram identificadas necessidades de novos experimentos. Ressalta-se que, conforme informado na seção 3, todas as versões de *dataset* foram balanceadas antes dos experimentos.

Na primeira fase de experimentos, o objetivo foi de identificar os algoritmos mais promissores e, para isso, buscou-se avaliar o desempenho dos modelos criados com diversos algoritmos, mas ainda com baixa variação de hiperparâmetros. Nesta fase, ainda não foram usados os atributos gerados pelos modelos NLP.

Os experimentos foram realizados usando-se validação cruzada com 10 *folds*, acurácia como métrica, *One-Hot Encoding* para transformar dados categóricos em numéricos, normalização e padronização para deixar os atributos numéricos na mesma ordem de grandeza e biblioteca *GridSearch* para auxiliar na variação de hiperparâmetros, mesmo que ainda de forma limitada.

Observa-se que, em um primeiro experimento, buscamos também validar a hipótese que o atributo "objeto" poderia ser excluído do *dataset*, visto que foi verificado que, por ser categórico e com alta variabilidade, ele gerava um número muito grande de novos atributos durante o *One-Hot Encoding*, deixando alguns experimentos extremamente lentos, como quando usamos o algoritmo KNN (*K-Nearest Neighbors* ou K-vizinhos mais próximos). Devido ao ganho marginal na acurácia do modelo, o atributo foi excluído do *dataset*.

Os algoritmos usados foram: KNN, Regressão Logística (*Logistic Regression*), SVM (*Support Vector Machines* ou Máquina de vetores de suporte), Árvore de decisão (*Classification and Regression Tree* - CART) e *Naive Bayes*, além dos métodos *ensembles Bagging* (com estimadores Árvore de Decisão e KNN), *Random Forests*, *Extra Trees* e *Gradient Boosting*. Para todos os *ensembles*, foram usados 10 estimadores.

As melhores acurácias foram obtidas com *Bagging*, *Random Forests*, *Extra Trees* e KNN, predominantemente com o pré-processamento *StandardScaller* (Padronização), sendo então as opções selecionadas para os experimentos seguintes. Os resultados são apresentados na Tabela 6.

Na fase seguinte, o objetivo foi realizar experimentos com os algoritmos selecionados na fase anterior, porém, com maior variação dos hiperparâmetros, além de se avaliar novas métricas de qualidade, como precisão, *recall* e *F1-score*.

Desta fase em diante, a métrica *recall* passou a ser a principal métrica para a avaliação do desempenho dos modelos, considerando as características do problema de negócio já citadas. No entanto, o comportamento das demais métricas também foi considerado na análise do desempenho.

Para os métodos *ensembles* foram usados 50 estimadores.

O melhor *recall* foi obtido pelo modelo *Bagging* com o estimador Árvore de Decisão, que também apresentou valores relativamente altos para as demais métricas.

O modelo KNN com distância de Manhattan e n = 21 apresentou o segundo *recall* mais alto, porém, seu treinamento foi extremamente lento. Acrescenta-se que, por se tratar de um algoritmo não paramétrico, que não gera uma função durante o treinamento, apresenta tempos muito altos de execução, a depender do *dataset*.

Table 7: Resultados da segunda fase de experimentos

Classificador	Hiperparâmetros para métrica Recall	Recall	Acurácia	Precisão	F1
Bag_DecTree_Knn	base_estimator: DecisionTreeClassifier, n_estimators: 50	92,16%	95,66%	85,76%	88,62%
KNN	metric: minkowski, n_neighbors: 5	91,34%	93,16%	77,19%	82,35%
RanFor	max_features: 9, n_estimators: 50	88,64%	95,92%	87,45%	88,18%
ExtTree	max_features: 9, n_estimators: 50	85,53%	95,13%	86,01%	85,71%

Table 8: Resultados da terceira fase de experimentos

Classificador	Hiperparâmetros para métrica Recall (n_estimators: 50)	Dataset	Recall	Acurácia	Precisão	F1
Bag_DecTree_Knn	base_estimator: DecisionTreeClassifier	LDA	93,04%	97,03%	87,57%	89,96%
Bag_DecTree_Knn	base_estimator: DecisionTreeClassifier	Mallet	92,33%	97,11%	88,04%	89,93%
Bag_DecTree_Knn	base_estimator: DecisionTreeClassifier	Kmeans	90,21%	93,03%	76,08%	80,90%
RanFor	max_features: 16	LDA	90,26%	97,00%	89,03%	89,35%
RanFor	max_features: 16	Mallet	90,06%	97,05%	88,82%	89,35%
RanFor	max_features: 16	Kmeans	84,57%	91,87%	72,69%	76,96%
ExtTree	max_features: 13	Kmeans	79,55%	89,12%	65,98%	70,06%
ExtTree	max_features: 13	LDA	87,51%	96,84%	88,03%	87,83%
ExtTree	max_features: 16	Mallet	86,42%	96,40%	86,82%	86,52%

Considerando este aspecto e a proximidade dos resultados dos modelos com melhor *recall*, foram selecionados, além do modelo com *Bagging* citado, os modelos *Randon Forests* com *max_features* = 9 e 50 estimadores e *Extra Trees* também com *max_features* = 9. Os resultados são apresentados na Tabela 7.

Na fase 3, o objetivo foi verificar se os melhores modelos encontrados na fase anterior conseguiriam ter desempenhos ainda melhores com os *datasets Kmeans*, *LDA* e *LDA Mallet*, gerados a partir dos diferentes modelos de NLP criados na etapa de pré-processamento do campo textual (atributo "texto longo"). Aqui, cabe observar que Antes de se iniciar os experimentos desta fase, foi necessário verificar se a mudança do *dataset*, incluindo-se os atributos gerados nos modelos NLP, poderia implicar em outros resultados na fase inicial de experimentos, onde os algoritmos mais promissores foram selecionados. Repetiu-se então esses experimentos com os novos *datasets* e observou-se que os algoritmos de melhor desempenho continuaram os mesmos. Destaca-se que, de acordo com o modelo de NLP, a quantidade de atributos gerados a partir do atributo "texto longo" foi diferente, com os *datasets Kmeans*, *LDA Gensim* e *LDA Mallet* apresentando 36, 23 e 15 atributos, respectivamente, considerando que o atributo "texto longo" foi substituído por diferentes quantidades de atributos no pré-processamento deste campo com as técnicas NLP, conforme explicado na seção 3.

Ao se analisar os resultados do *recall* nestes experimentos, observou-se que os modelos com o *ensemble Extra Trees* tiveram o pior desempenho. Os resultados são apresentados na Tabela 8. Pode-se observar que os modelos tiveram seu melhor desempenho com o *dataset LDA*, sendo então selecionados os modelos *Bagging* e *Random Forests* com o *dataset LDA* nesta fase.

Na fase 4, o objetivo foi realizar experimentos variando novos hiperparâmetros, porém, apenas para os dois modelos selecionados a partir dos resultados da fase anterior. Como os métodos *ensembles* têm execução bastante lenta, os experimentos se limitaram à variação dos hiperparâmetros *max_features* e *max_depth*, mantendo-se o número de estimadores = 50.

Observou-se que o modelo *Random Forests* com o parâmetro *max_features* = 0,5 teve o melhor desempenho, com métricas de *recall*, acurácia e precisão superiores ao melhor modelo com este *ensemble* na fase anterior. E o modelo *Bagging* com os parâmetros *max_features* = 0,5 e *max_depth* = 15 teve desempenho ligeiramente superior, em termos de *recall*, ao melhor modelo com este *ensemble* na fase anterior, conforme apresentado na Tabela 9.

Table 9: Resultados da quarta fase de experimentos

Classificador	Pré-processamento	Métrica	Parâmetros	Score	Dataset
Bag_DecTree	StandardScaler_OHE	Recall	max_depth: 15, max_features: 0.5	93,66%	LDA
RanFor	StandardScaler_OHE	Recall	max_features: 0.05, n_estimators: 50	93,35%	LDA
Bag_DecTree	StandardScaler_OHE	Precision	max_depth: 15, max_features: 0.2	84,95%	LDA
RanFor	StandardScaler_OHE	Precision	max_features: 0.05, n_estimators: 50	89,60%	LDA
Bag_DecTree	StandardScaler_OHE	F1	max_depth: 15, max_features: 0.2	88,39%	LDA
RanFor	StandardScaler_OHE	F1	max_features: 0.2, n_estimators: 50	90,99%	LDA
Bag_DecTree	StandardScaler_OHE	Accuracy	max_depth: 15, max_features: 0.5	96,11%	LDA
RanFor	StandardScaler_OHE	Accuracy	max_features: 0.5, n_estimators: 50	97,40%	LDA

Table 10: Resultados da última fase de experimentos - Testes

Classificador	Recall	Acurácia	Precisão	F1
BaggingClassifier(DecisionTree)	87,08%	95,19%	71,76%	78,68%
RandomForestClassifier	82,77%	96,16%	80,22%	81,47%

4.1 Testes e Últimos Resultados

Nesta última fase, foram realizados os experimentos com o *dataset* de teste, visando verificar a capacidade de generalização dos melhores modelos e selecionar aquele que será implantado.

Os resultados obtidos foram inferiores àqueles do treinamento, porém, foram considerados satisfatórios. Observou-se ainda uma melhor generalização do modelo com o *Bagging*. Os resultados são apresentados na Tabela 10.

Como o objetivo deste trabalho foi buscar o modelo com melhor *recall*, ainda que com bons resultados para as demais métricas analisadas, o modelo selecionado foi o ***Bagging*, com estimador Árvore de Decisão, 50 estimadores, *max_features* = 0.5 e *max_depth* = 15** que, adicionalmente, apresentou a melhor capacidade de generalização. Este modelo, então, será o proposto para uma primeira implantação. A Figura 4 mostra um fluxograma com as fases dos experimentos realizados.

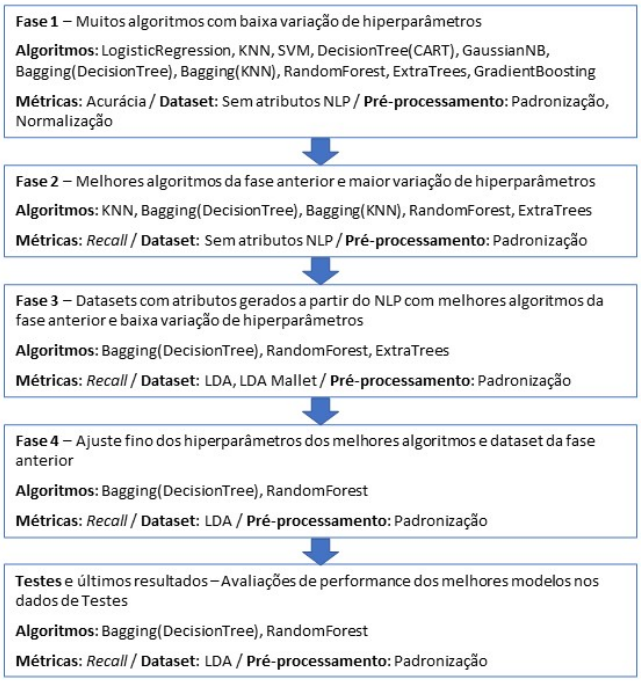

Figure 4: Fluxograma resumo das fases dos experimentos

5 CONCLUSÃO

Este trabalho apresentou o uso de técnicas de *machine learning* – algoritmos de classificação e métodos de processamento de linguagem natural (NLP) – para a construção de um modelo classificador de recomendações técnicas de inspeção, visando apoiar a equipe de manutenção na priorização das recomendações mais críticas e, assim, otimizar recursos, ganhar produtividade, aumentar a segurança operacional e evitar prejuízos decorrentes de paradas de produção.

Foram realizadas 5 fases de experimentos, onde lidou-se com diversos desafios técnicos, como um *dataset* desbalanceado e com predominância de atributos categóricos, necessidade de seleção de

métricas de qualidade mais adequadas ao problema e, principalmente, o tratamento de atributo textual livre de grande relevância, através de técnicas de NLP. Os experimentos foram realizados em fases, onde, inicialmente, buscou-se selecionar os algoritmos de melhor desempenho em um cenário de baixa variação de hiperparâmetros. Na sequência, avaliou-se o desempenho desses algoritmos com os *datasets* onde o campo textual foi pré-processado com técnicas de NLP. Por fim, concluiu-se com uma maior variação de hiperparâmetros, chegando-se, então, ao modelo de melhor desempenho, considerando o *recall* a principal métrica de avaliação.

O modelo de melhor desempenho utilizou-se de um *ensemble Bagging,* com estimador Árvore de Decisão, 50 estimadores, *max_features* = 0.5 e *max_depth* = 15, e o *dataset* onde o campo textual foi pré-processado com a técnica *LDA,* apresentando *recall* de 87,08% e acurácia de 95,19%, valores entendidos como promissores, considerando os desempenhos observados em trabalhos correlatos e a ausência de um método sistemático atualmente em uso pela equipe da unidade operacional em questão.

Como trabalhos futuros, pretende-se implantar o modelo em produção, avaliando-se o seu desempenho ao longo do tempo em um cenário de aplicação real. Adicionalmente, observa-se outras oportunidades de pesquisa, utilizando-se *datasets* maiores, com dados de outras unidades operacionais não avaliadas, outras técnicas de balanceamento e variações de novos hiperparâmetros, tanto no pré-processamento NLP, quanto nos modelos de classificação.

REFERENCES

[1] Instituto Nacional de Metrologia, Qualidade e Tecnologia-INMETRO. Portaria nº537, de 21 de outubro de 2015: Instrução normativa INMETRO para serviços próprios de inspeção de equipamentos - SPIE. Brasília: INMETRO, 2015.

[2] Çınar, Z.M., Abdussalam Nuhu, A., Zeeshan, Q., Korhan, O., Asmael, M. and Safaei, B. 2020. *Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0.* Sustainability. 12, 19 (Oct. 2020), 8211

[3] Lucchetti, G.P. *Aplicação de machine learning para predição de manutenção de máquinas em empresa de transporte e logística de combustíveis.* Trabalho de conclusão de curso (Engenharia de Produção). Niterói: Escola de Engenharia, Universidade Federal Fluminense. 2020.

[4] Machado, F.P. *Predição da hospitalização de pacientes idosos no departamento de emergência: abordagem utilizando aprendizado de máquina.* Dissertação (Mestrado profissional em Gestão para a Competitividade). São Paulo: Escola de Administração de Empresas de São Paulo, Fundação Getúlio Vargas. 2019.

[5] Moraes, M. Tomada de decisão na priorização de pacientes em fila de espera cirúrgica baseada em aprendizado de máquina. Dissertação (Mestrado profissional em Gestão para a Competitividade). São Paulo: Escola de Administração de Empresas de São Paulo, Fundação Getúlio Vargas. 2021.

[6] Escovedo, T. e Koshiyama, A.. Introdução a Data Science - Algoritmos de Machine Learning e métodos de análise. Casa do Código, 2020.

[7] Kalinowski, M.; Escovedo, T.; Villamizar, H.; Lopes, H. "Engenharia de Software para Ciência de Dados: Um Guia de Boas Práticas com Ênfase na Construção de Sistemas de Machine Learning em Python. Editora Casa do Código, 2023.