Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Predição de evasão de militares do Exército Brasileiro utilizando técnicas de machine learning

Marcella Guarnieri Mercês

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientador
Prof. Dr. Márcio de Carvalho Victorino

Coorientador
Dr. Wallace Anacleto Pinheiro

Brasília
2023

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

Mercês, Marcella Guarnieri
M554p Predição de evasão de militares do Exército Brasileiro utilizando técnicas de machine learning / Marcella Guarnieri Mercês; orientador Márcio de Carvalho Victorino; co-orientador Wallace Anacleto Pinheiro. -- Brasília, 2023.
74 p.

Dissertação(Mestrado Profissional em Computação Aplicada) - Universidade de Brasília, 2023.

1. Rotatividade de Empregados. 2. Aprendizado de Máquina. 3. Exército Brasileiro. I. Victorino, Márcio de Carvalho, orient. II. Pinheiro, Wallace Anacleto, co orient. III. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Predição de evasão de militares do Exército Brasileiro utilizando técnicas de machine learning

Marcella Guarnieri Mercês

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Prof. Dr. Márcio de Carvalho Victorino (Orientador)
Faculdade de Ciência da Informação

Dr. Wallace Anacleto Pinheiro
Coorientador

Prof. Dr. Thiago Paulo Faleiros
PPCA/UnB

Prof[a]. Dr[a]. Maria Cláudia Reis Cavalcanti
IME

Prof. Dr. Gladston Luiz da Silva
Coordenador do Programa de Pós-graduação em Computação Aplicada

Brasília, 22 de junho de 2023

Dedicatória

Dedico este trabalho à minha família, que sempre me apoiou e incentivou em tudo que fiz. Sem vocês eu não teria chegado tão longe. Obrigada!

Agradecimentos

Primeiramente, agradeço à minha família. Aos meus pais, Vânia e Carlos Eduardo, por sempre terem me mostrado o valor do estudo; à minha irmã, Luísa, pela amizade e palavras de incentivo; e ao meu marido, Willian, que sempre me apoiou e acreditou em mim mais do que eu mesma.

Agradeço ao meu orientador, professor doutor Márcio Victorino, e ao meu coorientador, doutor Wallace Pinheiro, pela motivação, profissionalismo e apoio no desenvolvimento desta pesquisa, sem os quais ela não teria sido concluída com êxito.

Agradeço à colega Beatriz Fragnan, pelo apoio nos estudos e no trabalho e por sempre me incentivar a seguir em frente com a pesquisa.

Por fim, agradeço à Universidade de Brasília pela oportunidade de realizar este curso tão conceituado. Em especial, ao Programa de Pós-Graduação em Computação Aplicada e aos seus professores.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Esse trabalho teve por objetivo realizar uma análise do quadro demissionário de oficiais do Exército Brasileiro ao longo dos anos e propor um conjunto de técnicas que possibilitem identificar os oficiais com maior probabilidade de se tornarem demissionários, de forma a permitir que o Exército aja proativamente a fim de evitar as perdas que esse ato causa para a Força Terrestre. As abordagens escolhidas levaram em consideração as técnicas de aprendizado de máquina mais utilizadas para a predição de demissão voluntária de empregados em empresas dos mais diversos ramos de atuação, como os algoritmos de classificação K-nearest neighbors (KNN), árvores de decisão, random forest, gradient boosting, extreme gradient boosting (XGBoost) e CatBoost. Também foram testadas a utilização de técnicas de detecção de *outliers* e a utilização dos algoritmos de classificação em conjunto com técnicas de reamostragem, combinando técnicas de *undersampling* com técnicas de *oversampling*. O estudo de caso foi feito de acordo com o *framework* CRISP-DM e utilizou dados de oficiais do Exército Brasileiro que ingressaram nessa Força entre 1990 e 2020. Todas as combinações de técnicas e algoritmos foram analisadas para o conjunto de dados completo e para conjuntos de dados separados por carreiras.

Foi possível obter resultados de predição satisfatórios para o objetivo do estudo, com uma predição superior à obtida através de um algoritmo que seleciona apenas a classe mais comum. Três algoritmos utilizados no conjunto de dados completo obtiveram o melhor resultado, sendo estatisticamente equivalente entre eles: XGBoost e CatBoost com a realização de etapas de pré-processamento e CatBoost sem a realização de etapas de pré-processamento. A separação por carreira não foi capaz de melhorar o resultado da predição obtida com o conjunto de dados completo. Para a implantação, foi escolhido o resultado do CatBoost sem as etapas de pré-processamento, por ser o mais rápido em treinamento e mais simples em implementação entre os melhores resultados. Os atributos com maior importância para essa classificação foram a quantidade de cursos, o tempo de serviço e o tempo desde formado.

Palavras-chave: Rotatividade de Empregados, Classificação, Aprendizado de Máquina, Predição de Rotatividade

Abstract

This research aims to analyze the turnover of Brazilian Army's officers over the years and propose a set of techniques that allow the identification of those officers more likely to resign. This result can help the Brazilian Army to act so that these resignations have a lesser impact in it's productivity. The approaches chosen considered the most common techniques used to predict employee churn in other areas. Common classification algorithms, such as K-nearest neighbors (KNN), decision trees, random forest, gradient boosting, extreme gradient boosting (XGBoost), and CatBoost, were used. Techniques to detect outliers in the dataset were also tested, as well as resampling techniques with the combination of both undersampling and oversampling algorithms. The case study used the framework CRISP-DM. The chosen data set had the data of all officers that entered the Brazilian Army between 1990 and 2020. All techniques and algorithms combinations were tested for both the whole data set and for data sets divided by the different careers officers can have.

Satisfactory results were achieved by obtaining a model that is better at predicting turnover than an algorithm that always classifies all elements as the majority class. In fact, three of the models obtained presented the best results, with all three being statistically equivalent to each other: XGBoost and CatBoost combined with data pre-processing and CatBoost without any data pre-processing. Dividing the data by career did not improve the results obtained with the whole data set. For the implantation part of CRISP-DM the model generated with CatBoost without data pre-processing was chosen, since it is the one that has faster training and simpler code among the three best. It was also possible to discover that the attributes with the highest importance to the classification were number of courses, time since joining the Army and time since graduating.

Keywords: Employee Churn, Classification, Machine Learning, Churn Prediction

Sumário

1 Introdução **1**
1.1 Definição do Problema 1
1.2 Justificativa 2
1.3 Objetivo Geral 3
1.4 Objetivos Específicos 4
1.5 Estrutura do Trabalho 4

2 Procedimento Metodológico **5**
2.1 Teoria do Enfoque Meta Analítico Consolidado - TEMAC 5
2.1.1 Preparação da Pesquisa 5
2.1.2 Apresentação e interrelação dos dados 6
2.1.3 Detalhamento do modelo integrador e validação por evidências . . . 8
2.1.4 Seleção final dos artigos de trabalhos relacionados 10
2.2 Classificação da Pesquisa 11
2.3 CRISP-DM 12
2.3.1 Entendimento do Negócio 12
2.3.2 Entendimento dos Dados 13
2.3.3 Preparação dos Dados 13
2.3.4 Modelagem 13
2.3.5 Avaliação 13
2.3.6 Implantação 14

3 Trabalhos Relacionados **15**
3.1 Artigos mais recentes de revisão da literatura 15
3.2 Estudos de caso sobre predição da saída de funcionários 16
3.3 Abordagens relacionadas em outras áreas de estudo 18
3.4 Quadro resumo dos trabalhos relacionados 19

4 Fundamentação Teórica **21**
4.1 O problema de rotatividade de empregados 21

4.2 Pré-processamento dos dados . . . 22
4.2.1 Transformação dos atributos categóricos . . . 23
4.2.2 Normalização dos atributos numéricos . . . 24
4.3 Algoritmos de classificação . . . 26
4.3.1 K-Nearest Neighbors (KNN) . . . 26
4.3.2 Árvores de Decisão . . . 27
4.3.3 Random Forest . . . 29
4.3.4 Gradient Boosting . . . 30
4.4 Algoritmos de reamostragem para conjuntos de dados desbalanceados . . . 31
4.4.1 SMOTE . . . 31
4.4.2 ADASYN . . . 33
4.4.3 Near Miss . . . 33
4.4.4 Tomek Link . . . 34
4.4.5 Edited Nearest Neighbors . . . 34
4.5 Métodos de detecção de *outliers* . . . 34
4.5.1 Local Outlier Factor (LOF) . . . 35
4.5.2 Isolation Forest . . . 35
4.6 Teste de McNemar para comparação de modelos de aprendizado de máquina 36

5 Estudo de Caso 38
5.1 Entendimento do Negócio . . . 38
5.2 Entendimento dos Dados . . . 39
5.3 Preparação dos Dados . . . 41
5.3.1 Obtenção do conjunto de dados . . . 41
5.3.2 Pré-processamento dos dados . . . 44
5.4 Modelagem . . . 48
5.4.1 Primeiro grupo de testes . . . 49
5.4.2 Segundo grupo de testes . . . 51
5.4.3 Terceiro grupo de testes . . . 52
5.5 Avaliação . . . 52
5.5.1 Conjunto de dados completo . . . 54
5.5.2 Conjunto de dados separado por carreira . . . 62
5.6 Implantação . . . 64

6 Conclusão 67

Referências 70

Lista de Figuras

2.1 Quantidade de artigos sobre o tema publicados desde 2016 até 2021 no *Web of Science* 7
2.2 Quantidade de artigos sobre o tema publicados desde 2014 até 2021 no *Scopus* 8
2.3 Análise de co-citação nos artigos do *Web of Science* 9
2.4 Análise de acoplamento bibliográfico nos artigos do *Web of Science* 10
2.5 Nuvem de palavras com as principais palavras-chave dos artigos 10
2.6 Fases da CRISP-DM [1] 12

4.1 Exemplo do funcionamento do One Hot Encoding 23
4.2 Exemplo do funcionamento do Target Encoding 24
4.3 Exemplo do funcionamento do Leave-one-out Encoding 25
4.4 Separação dos dados em quartis 26
4.5 Exemplo de classificação com KNN [2] 27
4.6 Exemplo de classificação com árvore de decisão [3] 28

5.1 Evasão de oficiais do Exército por turma de formação. 40
5.2 Evasão de oficiais do Exército por tempo de serviço e posto 40
5.3 Evasão de oficiais do Exército por Quadro, Arma ou Serviço 41
5.4 Importâncias dos atributos na classificação realizada pelo CatBoost sem pré-processamento 60
5.5 Importância dos atributos na classificação realizada pelo CatBoost sem pré-processamento com método SHAP Values 61
5.6 Importância dos atributos na classificação realizada pelo CatBoost sem pré-processamento por carreira 65
5.7 Esquema de treinamento e implantação do melhor modelo encontrado e do de como novos dados serão incorporados. 66

Lista de Tabelas

3.1 Quadro resumo dos trabalhos relacionados. 19

4.1 Tabela de contingência a ser utilizada para o teste de McNemar 37

5.1 Atributos selecionados com sua descrição e seus tipos de dados. 45
5.2 Atributos selecionados para passar pela codificação. 46
5.3 Parâmetros testados em cada algoritmo de codificação. 47
5.4 Parâmetros testados para cada classificador. 50
5.5 Parâmetros testados para cada algoritmo de reamostragem. 51
5.6 Parâmetros testados para cada algoritmo de detecção de outliers. 52
5.7 Acurácia do DummyClassifier e quantidade de elementos em cada classe pra o conjunto de dados completo e por carreira. 53
5.8 Métricas calculadas para cada classificador 54
5.9 Melhores parâmetros e pré-processamento para cada classificador 55
5.10 Métricas calculadas para cada classificador com a utilização de um método de reamostragem 56
5.11 Melhores parâmetros e pré-processamento para cada classificador com a utilização de um método de reamostragem 57
5.12 Métricas calculadas para cada algoritmo de detecção de *outliers* 57
5.13 Melhores parâmetros e pré-processamentos para cada algoritmo de detecção de *outliers* 58
5.14 Resumo dos resultados apresentados pelos melhores modelos encontrados . 58
5.15 Predições do conjunto de validação separadas por qualificação 62
5.16 Predições do conjunto de validação separadas por carreiras 63
5.17 Métricas obtidas com a execução do CatBoost sem pré-processamento nos conjuntos de dados separados por carreiras 64

Lista de Abreviaturas e Siglas

AMAN Academia Militar das Agulhas Negras.

AUC área sobre a curva ROC.

BI Business Intelligence.

CBLOF cluster-based local outlier factors.

CCOp Mv Centro de Coordenação de Operações Móvel.

CRISP-DM Cross Industry Standard Process for Data Mining.

DAPROM Diretoria de Avaliação e Promoções.

DCEM Diretoria de Controle de Efetivos e Movimentações.

DCT Departamento de Ciência e Tecnologia.

DGP Departamento-Geral do Pessoal.

EB Exército Brasileiro.

EME Estado-Maior do Exército.

GCN graph convolutional networks.

KNN k-nearest neighbors.

LOF local outlier factor.

LSTM long short-term memory.

LTIP Licença para Tratar de Interesse Particular.

OM Organização Militar.

OMDS organizações militares diretamente subordinadas.

PNR Próprio Nacional Residencial.

QAO Quadro Auxiliar de Oficiais.

QCO Quadro Complementar de Oficiais.

QEM Quadro de Engenheiros Militares.

QMB Quadro de Material Bélico.

SVM support vector machines.

TEMAC Teoria do Enfoque Meta Analítico Consolidado.

Capítulo 1

Introdução

O projeto do Centro de Coordenação de Operações Móvel (CCOp Mv) do Exército Brasileiro (EB) tem como objetivo geral contribuir para a ampliação da capacidade de planejamento e coordenação da Força Terrestre em operações na proteção da sociedade. Ele se alinha à missão constitucional das Forças Armadas de defender a Pátria, determinada no Artigo nº 142 da Constituição Federal de 1988, e à Estratégia Nacional de Defesa, que prioriza a capacidade dissuasória da Força por meio da utilização de recursos tecnológicos combinados com a adequada preparação das estruturas operativas e a capacitação de recursos humanos. Nesse contexto, a existência de militares devidamente qualificados para operar os produtos tecnológicos e participar das missões a serem empreendidas com os CCOp Mv é de extrema importância para garantir que o emprego destes centros seja bem sucedido. O problema de evasão de oficiais que ocorre no Exército Brasileiro afeta diretamente nessa questão, com grande potencial de afetar esse e outros projetos estratégicos dessa Força Armada.

1.1 Definição do Problema

Ao longo dos últimos anos, o Exército vem perdendo parte de seu pessoal qualificado para o mercado de trabalho civil, afetando diretamente sua capacidade de planejar e executar com sucesso sua missão constitucional. Com isso em mente, o Departamento-Geral do Pessoal (DGP), organização militar responsável pelo gerenciamento dos recursos humanos do Exército, e o Estado-Maior do Exército (EME), órgão responsável pelo planejamento estratégico do Exército, confeccionaram, com o apoio de uma ferramenta de Business Intelligence (BI), diversos relatórios para analisar a situação de pedidos de demissão. Dentre todos os militares do Exército, os oficiais são o grupo que tem um maior nível de capacitação e também um maior tempo de formação. Analisando os dados dos graduados entre 1990 e 2020, esses relatórios mostram que 19.781 pessoas concluíram algum curso

de formação de oficiais, das quais 1.541 pediram demissão antes do fim da carreira, o que representa uma taxa de evasão de 7,79%.

Apesar da análise dos dados de evasão do Exército por estatística descritiva fornecer aos gestores do Exército uma boa visão do histórico de pedidos de demissão, até o momento não houve iniciativa com o intuito de aplicar técnicas de ciência de dados sobre os dados de militares do Exército para prever o quantitativo de militares que vão evadir a cada ano ou para identificar quem serão esses militares.

1.2 Justificativa

A perda de profissionais é problemática para qualquer empresa. O desempenho e produtividade dos empregados não depende apenas de seu conhecimento técnico, mas também de sua integração com os outros membros da equipe e seu conhecimento dos processos e do negócio da empresa, os quais só são aprendidos com o tempo [4]. Dessa forma, um empregado que pede demissão levará consigo um conhecimento que não é possível ser passado para quem vai substituí-lo e o novo contratado levará um tempo até ser capaz de apresentar produtividade similar, independentemente de quanto conhecimento técnico essa pessoa tenha. Além disso, há ainda os custos com treinamentos e outros investimentos que a empresa fez no funcionário que saiu e que terá que fazer novamente para o novo contratado. Apesar do custo, empresas civis têm a possibilidade de contratar profissionais com a qualificação técnica que elas precisam, por mais alta que ela seja. Se o empregado que sai tinha nível de doutorado, a empresa consegue contratar outra pessoa com conhecimento equivalente, ainda que o processo de encontrar esse profissional seja custoso.

No Exército, o impacto da perda de um profissional é ainda maior do que em uma empresa civil, uma vez que não é possível contratar um profissional com o mesmo conhecimento técnico do anterior. Para ingressar na carreira militar, o profissional deverá sempre começar do início dela e ir progredindo aos poucos. Os cursos e avanços da carreira quase sempre estão ligados ao tempo de Exército que o militar tem, não sendo possível acelerar esse processo nem mesmo pelas necessidades da Força. Se, por exemplo, um capitão de infantaria pedir demissão, ninguém a mais será promovido para suprir essa vaga em aberto. O Exército terá que realocar os capitães de infantaria existentes para ocupar as vagas de acordo com a relevância das funções para o Exército, mas as vagas com menor importância ou ficarão em aberto ou serão ocupadas por alguém com menos qualificação do que o pretendido para a função. Conforme a carreira dos capitães que ficaram for avançando, sempre haverá uma pessoa a menos do que o pretendido inicialmente pelo Exército quando as vagas de ingresso na Academia Militar das Agulhas Negras

(AMAN) foram decididas. Além disso, não é possível para o Exército simplesmente abrir uma quantidade maior de vagas para novos militares considerando que alguns daqueles militares sairão no futuro, uma vez que a quantidade de pessoas em cada especialização e posto é limitada por uma lei alterada anualmente. Quando os quantitativos nesta lei não são corretamente estimados, ela pode impedir o atendimento das necessidades de pessoal mais capacitados nos postos mais altos, gerando prejuízos para a instituição.

Nesse contexto, percebe-se que ter uma previsão de quais militares sairão do Exército ao longo do tempo tem grande potencial de diminuir o impacto das demissões. Uma informação mais precisa de quem vai pedir demissão pode ajudar no planejamento das vagas, garantindo um maior aproveitamento das pessoas sem ultrapassar os limites estabelecidos pela lei do efetivo do Exército. Também pode permitir que o preparo do substituto daquela pessoa, que já se sabe que será um potencial demissionário, se inicie antes que esse pedido de demissão ocorra de fato, preparando um profissional substituto em um tempo menor. Por fim, obter o conhecimento da possibilidade de evasão com uma maior antecedência pode permitir que o Exército tome medidas que efetivamente evitem essa demissão.

Uma ferramenta capaz de realizar esse tipo de previsão pode beneficiar diversas organizações militares do Exército, uma vez que cada uma é responsável por partes diferentes da vida do militar. Um dos beneficiados certamente será o Departamento-Geral do Pessoal (DGP) e suas organizações militares diretamente subordinadas (OMDS), entre as quais encontram-se os órgãos responsáveis por cursos, movimentações e promoções dos militares. Também serão beneficiados o Departamento de Ciência e Tecnologia (DCT), responsável pela carreira dos militares das áreas de tecnologia do Exército, o Estado-Maior do Exército (EME), que é responsável por realizar o planejamento estratégico do Exército e definir as diretrizes que vão orientar o trabalho dos demais órgãos, entre outros.

1.3 Objetivo Geral

Diante do exposto, esse trabalho tem por finalidade analisar a evolução do quadro demissionário do EB ao longo dos anos e propor um conjunto de técnicas que possibilitem identificar oficiais com maior probabilidade de se tornarem demissionários, de forma a permitir que o Exército aja proativamente a fim de evitar as perdas que esse ato causa para a Força Terrestre e, mais especificamente, para evitar que demissões prejudiquem o emprego dos CCOp Mv.

1.4 Objetivos Específicos

Para alcançar o objetivo proposto, definiu-se os seguintes objetivos específicos:

- Detalhar o problema da evasão de oficiais do Exército Brasileiro por meio do uso de ferramentas de *Business Intelligence* já utilizadas por essa Força para entender, por meio de estatística descritiva, a situação atual dessa evasão;
- Encontrar uma combinação de algoritmos e métodos para prever a evasão de oficiais do Exército tanto de forma geral quanto realizando recortes por Arma, Quadro e Serviço.
- Obter um padrão para a evasão dos oficiais.

1.5 Estrutura do Trabalho

No Capítulo 2, será explicada a metodologia dessa pesquisa, a qual constituiu na revisão sistemática da literatura através do TEMAC para as etapas mostradas nos capítulos 3 e 4 e na utilização do CRISP-DM para o desenvolvimento do estudo de caso.

No Capítulo 3, serão abordados os principais trabalhos relacionados a essa pesquisa, as técnicas utilizadas por eles e os principais resultados encontrados.

No Capítulo 4, será apresentada a fundamentação teórica desse trabalho, com os principais conceitos utilizados para o desenvolvimento da pesquisa. Entre os tópicos discutidos estão o problema de retenção, técnicas utilizadas para preparar os dados para utilizar numa classificação, algoritmos de classificação, métodos para lidar com dados desbalanceados e métodos de detecção de *outliers*.

O Capítulo 5 aborda o estudo de caso definido para obter a previsão de evasão de oficiais do Exército Brasileiro e o que foi feito em cada uma das etapas do CRISP-DM.

O Capítulo 6 apresenta a conclusão dessa dissertação, com os principais resultados obtidos e contribuições feitas.

Capítulo 2

Procedimento Metodológico

Esta pesquisa utiliza como metodologia a revisão sistemática da literatura, tanto para encontrar os principais estudos relacionados, mencionados no capítulo três, quanto para entendimento dos tópicos da fundamentação teórica apresentada no capítulo quatro. Essa revisão sistemática utilizou a Teoria do Enfoque Meta Analítico Consolidado (TEMAC), de Mariano e Rocha [5]. A partir da revisão sistemática, foi realizado um estudo de caso com o *framework* Cross Industry Standard Process for Data Mining (CRISP-DM).

2.1 Teoria do Enfoque Meta Analítico Consolidado - TEMAC

2.1.1 Preparação da Pesquisa

Na primeira etapa, são definidos os termos da pesquisa a ser realizada nas bases científicas selecionadas. Foram escolhidas as bases científicas *Web of Science* e *Scopus* devido à relevância que elas têm na comunidade acadêmica, à abrangência de assuntos que são retornados e à facilidade na aplicação de filtros na pesquisa. Os termos utilizados para a pesquisa em cada base e a quantidade de resultados encontrados em setembro de 2021 foram os seguintes:

1. Web of Science: 1.414 resultados para ALL=((employee OR "human resources"OR HR OR labour OR personnel OR worker OR workforce) AND (turnover OR churn OR retention OR attrition) AND (prediction OR forecast OR projection OR predictive OR "machine learning"OR "deep learning"OR "neural network"))
2. Scopus: 1.607 resultados para TITLE-ABS-KEY((employee OR "human resources"OR HR OR labour OR personnel OR worker OR workforce) and (turnover

or churn or retention or attrition) and (prediction OR forecast OR projection OR predictive OR "machine learning"OR "deep learning"OR "neural network"))

Para chegar nos termos de pesquisa acima, iniciou-se com as palavras mais conhecidas para tratar do problema, "*employee churn*". Observando os resultados dessa busca foi possível detectar sinônimos para esses vocábulos, os quais foram adicionados à busca para expandir os resultados. Esse procedimento foi repetido até não se encontrar mais nenhum termo amplamente utilizado para tratar do assunto.

A utilização de diversos termos sinônimos permitiu que uma quantidade considerável de artigos fosse encontrada, evitando que artigos de interesse não estivessem na lista de resultados. Porém, percebeu-se que muitos artigos de assuntos não relacionados ou com abordagens focando em áreas muito diferentes das de interesse também estavam nos resultados. Portanto, julgou-se necessário que os resultados da etapa anterior passassem por uma filtragem de temas e, a seguir, por uma filtragem manual que selecionou os trabalhos em que o título e o resumo indicavam que o artigo trata da predição, através de métodos computacionais, de demissão voluntária de funcionários em qualquer área, ou que tratavam da motivação para essa demissão. Foram excluídos nessa filtragem trabalhos que analisavam o problema de um ponto de vista psicológico ou de um profissional de recursos humanos, bem como estudos que não tinham relação com o tópico de demissão voluntária de funcionários. Por outro lado, foram mantidas pesquisas de outras áreas que não a computação, mas que eram relacionadas a recursos humanos em organizações militares.

Após a filtragem, adicionou-se aos resultados os artigos citados pelos encontrados na primeira busca. A nova lista de artigos passou novamente por uma filtragem manual dos artigos de interesse para o tópico pesquisado e o resultado encontrado será analisado de acordo com as etapas do TEMAC nas próximas seções.

2.1.2 Apresentação e interrelação dos dados

A exportação de dados de diferentes bases dificulta a junção dos resultados para uma análise unificada. Dessa forma, a análise sobre os artigos foi feita para os resultados encontrados em cada base separadamente.

No *Web of Science*, após as filtragens, restaram 124 artigos. Os principais autores que publicaram sobre o assunto, com ao menos três artigos publicados, foram:

- H. Xiong - 5 artigos
- H.S. Zhu - 5 artigos
- M.J. Somers - 3 artigos

Os países com maior número de publicações e com ao menos quatro artigos publicados foram:

- Estados Unidos da América - 42 artigos
- China - 20 artigos
- Índia - 19 artigos
- Taiwan - 5 artigos
- Canadá - 4 artigos
- África do Sul - 4 artigos

O Brasil aparece na lista com apenas um artigo publicado.

Com relação ao ano de publicação dos artigos, tem-se registros desde a década de 1970. No entanto, a grande maioria se concentra nos últimos anos. No gráfico exportado da *Web of Science*, apresentado na figura 2.1, é possível ver que o número de artigos sobre o tema tem crescido bastante ultimamente.

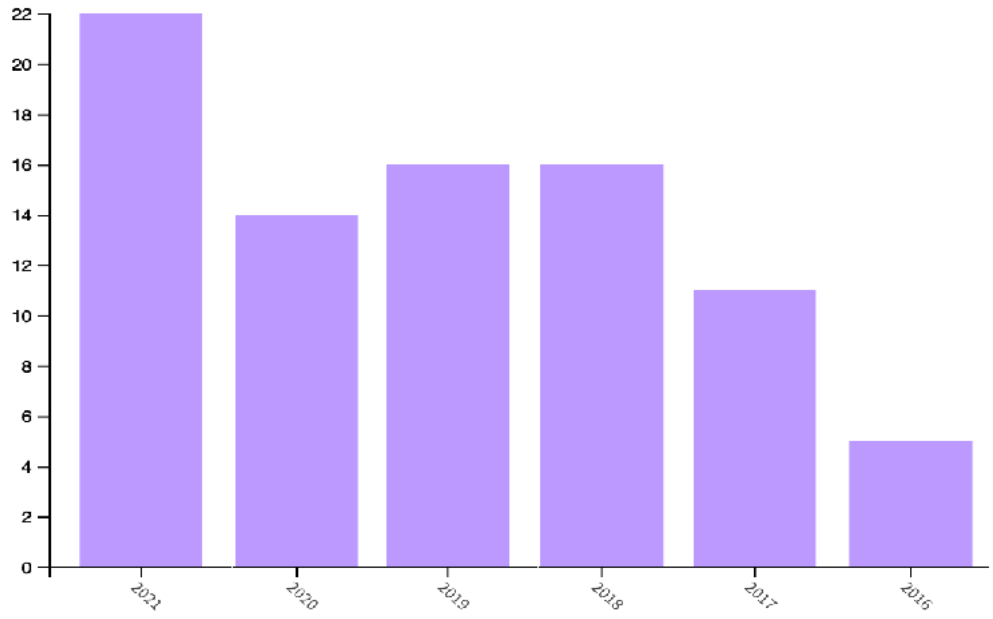


Figura 2.1: Quantidade de artigos sobre o tema publicados desde 2016 até 2021 no *Web of Science*

No *Scopus*, após as filtragens, restaram 154 artigos. Apenas um autor, T. R. Mitchell, publicou três artigos entre esses resultados, com outros autores publicando dois ou menos.

Os países com maior número de publicações foram:

- Estados Unidos da América - 44 artigos

- Índia - 30 artigos
- China - 16 artigos
- Reino Unido - 6 artigos
- Austrália - 5 artigos
- Canadá - 4 artigos

O Brasil aparece na lista com três artigos.

Com relação ao ano de publicação percebe-se um comportamento similar ao encontrado no *Web of Science*. A figura 2.2 mostra o gráfico do *Scopus* com o número de publicações anuais sobre o tema nos últimos sete anos. Novamente, é possível observar uma tendência de crescimento.

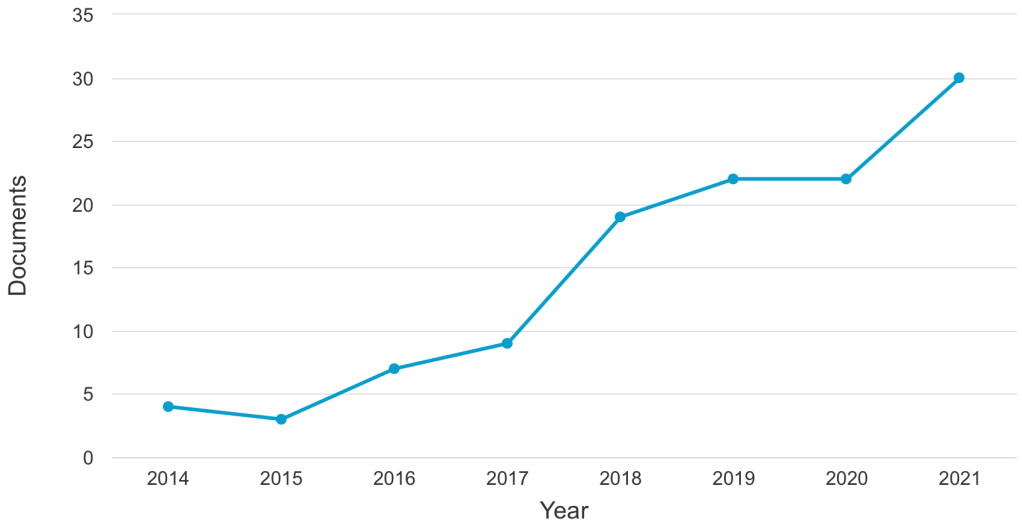


Figura 2.2: Quantidade de artigos sobre o tema publicados desde 2014 até 2021 no *Scopus*

2.1.3 Detalhamento do modelo integrador e validação por evidências

Na terceira etapa do TEMAC, é feita a análise de alguns índices bibliométricos. Foram selecionadas as análises de co-citação e de acoplamento bibliográfico e a frequência de palavras-chave. Na análise de co-citação, são encontrados os artigos comumente citados juntos por um mesmo artigo, o que pode ser considerado um indicativo de se tratarem de artigos sobre o mesmo tema. Já no acoplamento bibliográfico, são analisados quais artigos têm citações em comum, o que ajuda na identificação de possíveis frentes de pesquisas dentro do assunto [5]. A frequência de palavras-chave, como o próprio nome indica, mostra as principais palavras-chave utilizadas pelos artigos.

Para as duas primeiras análises, foi utilizado um *software* chamado VOSViewer, o qual aceita dados exportados dos relatórios de citações disponibilizados pelas principais bases de pesquisa, incluindo *Scopus* e *Web of Science*. Infelizmente, o formato de exportação dessas bases é diferente, impossibilitando a junção dos resultados. Assim, os dados obtidos nas duas bases também serão analisados separadamente nessa etapa. Será mostrada a análise feita para os resultados do *Web of Science*.

Na figura 2.3, é possível ver os três *clusters* obtidos com a análise de co-citação dos artigos obtidos no *Web of Science*. Analisando os itens de cada *cluster*, percebe-se que o representado em verde contém apenas artigos anteriores a 1980 e todos estão em publicações de psicologia. O *cluster* em vermelho tem tanto trabalhos mais antigos quanto trabalhos mais novos e o ponto em comum entre os artigos é serem publicados em periódicos e conferências sobre gerência de recursos humanos. Por fim, o *cluster* em azul mostra os artigos da área de computação, todos posteriores ao ano 2000. Pelas ligações, é possível ver que mesmo a abordagem computacional do assunto utiliza como referência as primeiras pesquisas feitas sobre a evasão de funcionários de empresas.

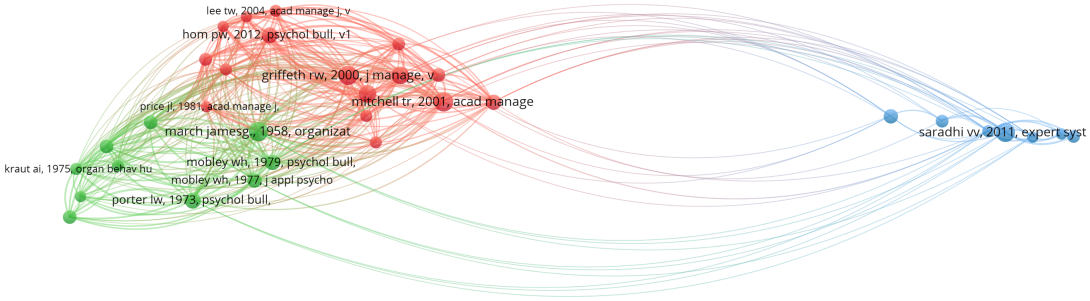


Figura 2.3: Análise de co-citação nos artigos do *Web of Science*

Para a análise de acoplamento bibliográfico, como o objetivo é encontrar as linhas de pesquisa seguidas, é comum considerar apenas os artigos publicados nos últimos três anos. A figura 2.4 mostra os quatro *clusters* encontrados.

Para obter a frequência de palavras-chave, as palavras-chave dos artigos encontrados nas duas bases foram misturadas sem retirada dos artigos duplicados. O resultado pode ser visto na figura 2.5. Observa-se que, apesar da pesquisa ter se iniciado pelos termo *churn*, o sinônimo *turnover* é muito mais utilizado, enquanto o termo *attrition* é tão utilizado quanto *churn*. Com isso, percebe-se que adicioná-los nos termos de busca certamente teve impacto positivo na quantidade de resultados encontrados.

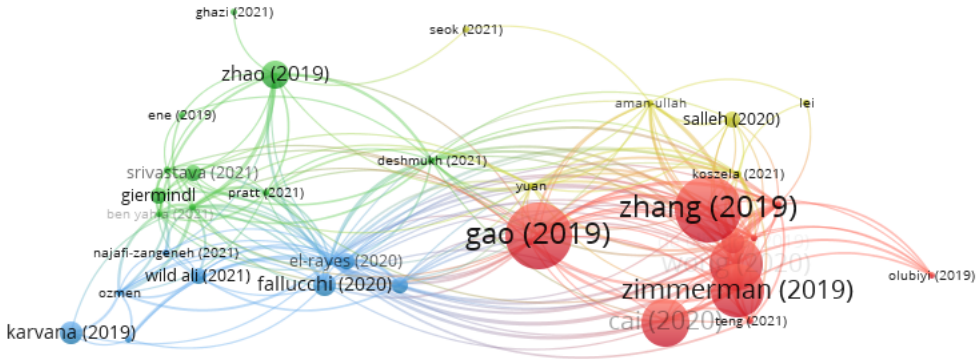


Figura 2.4: Análise de acoplamento bibliográfico nos artigos do *Web of Science*

Figura 2.5: Nuvem de palavras com as principais palavras-chave dos artigos

2.1.4 Seleção final dos artigos de trabalhos relacionados

A análise dos resultados feita até o momento considerou os resultados encontrados nas duas bases pesquisadas de forma separada. No entanto, para realizar a seleção de artigos relacionados ao tema do trabalho para serem analisados no próximo capítulo, foi necessário realizar a junção dos resultados da pesquisa feita nas duas bases para retirar os artigos repetidos, o que totalizou 244 referências distintas.

Analisando esses resultados, percebeu-se que tentar prever a saída voluntária de um empregado não é um interesse recente. A revisão da literatura sobre o assunto feita por Muchinsky e Tuttle em 1979 [6] menciona as principais formas existentes naquele momento

para tentar prever quais funcionários deixariam a empresa. Os trabalhos analisados por eles foram publicados entre as décadas de 1920 e 1970 e tentam prever a demissão de empregados com a análise de aspectos psicológicos a partir de testes de personalidade e inteligência, e também com aspectos da vida privada, como tamanho da família, estado civil, sexo e se a pessoa tem a principal renda da família ou não.

Apesar das diversas mudanças que o tempo trouxe tanto para o mercado de trabalho quanto para a tecnologia, o interesse na predição da saída de empregados seguiu existindo. Por muitos anos, os estudos sobre o assunto seguiram como domínio das áreas de psicologia e de gerência de recursos humanos, mas, nas últimas duas décadas, eles também passaram a aparecer com frequência nas pesquisas de computação. Dos 244 artigos distintos selecionados com o método TEMAC, mais da metade foi publicada nos últimos cinco anos e a maior parte deles utilizou algum método computacional para fazer a predição.

Com o objetivo de entender quais as técnicas utilizadas nas pesquisas mais recentes sobre a demissão voluntária de empregados, foram selecionados, entre os resultados da pesquisa, 16 artigos que serão analisados no próximo capítulo. Os critérios de seleção utilizados foram a data de publicação, priorizando artigos mais recentes, as técnicas de mineração utilizadas, buscando artigos que utilizassem abordagens mais diversas, a relevância dos autores na área, o que foi definido através da quantidade de citações dos autores e das análises de co-citação e acoplamento bibliográfico nas 244 referências selecionadas, e se a pesquisa se relacionava ao meio militar ou ao funcionarismo público de algum país, uma vez que esses ambientes têm características específicas e julgou-se interessante analisar como outros pesquisadores já lidaram com essas particularidades em suas pesquisas.

2.2 Classificação da Pesquisa

Métodos de pesquisa recebem classificações para facilitar a identificação, por parte dos pesquisadores, de qual tipo de pesquisa é mais adequado para resolver um problema [7] e para facilitar a definição das etapas a serem seguidas. Os métodos podem ser classificados quanto à sua natureza, objetivo ou procedimentos técnicos [8].

Nesta pesquisa, a natureza é aplicada, pois pretende obter conhecimento para a solução de um problema específico [7]. Quanto ao objetivo, esta pesquisa é exploratória, assim como todo estudo de caso [7], pois pretende examinar um fenômeno, no caso, a evasão de oficiais do Exército Brasileiro, para obter mais familiaridade sobre ele. Por fim, quanto ao procedimento metodológico, esta pesquisa se enquadra no estudo de caso por estudar um fenômeno individual que se destaca dos demais [9], mas também pode se enquadrar como

pesquisa bibliográfica, uma vez que uma metodologia foi seguida para obter as referências bibliográficas utilizadas como embasamento do estudo de caso.

2.3 CRISP-DM

O presente trabalho fez uso do estudo de caso como estratégia de pesquisa utilizando o *framework* Cross Industry Standard Process for Data Mining (CRISP-DM), composto por seis fases, as quais podem ser vistas na figura 2.6.

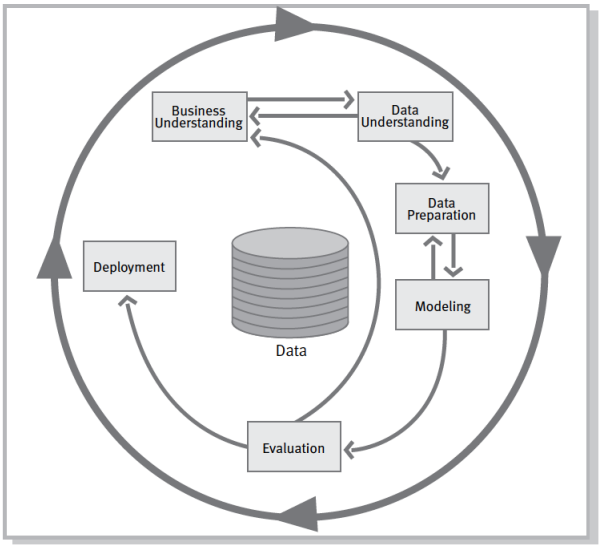


Figura 2.6: Fases da CRISP-DM [1]

Apesar da figura apresentar setas entre etapas específicas do processo, na realidade a sequência entre elas não é rígida e a qualquer momento pode ser decidido retornar para qualquer uma das etapas já executadas dependendo dos resultados obtidos. A seguir, será fornecida uma breve explicação de cada fase desse *framework* de acordo com as explicações apresentadas em seu guia de usuário [1].

2.3.1 Entendimento do Negócio

Nessa etapa do CRISP-DM o pesquisador deve entender o problema a ser resolvido, os objetivos e requisitos do ponto de vista do negócio. A partir desse entendimento, o problema do negócio deve ser traduzido para um problema de aprendizado de máquina, ou seja, descrito com termos técnicos e tendo objetivos claros que sejam relacionados aos objetivos do negócio.

2.3.2 Entendimento dos Dados

A etapa seguinte consiste em obter o acesso às fontes de dados necessárias para resolver o problema. Uma coleta inicial de dados deve ser feita para entender o conteúdo dessas fontes e qual a qualidade deles. Os dados podem ser visualizados através de gráficos ou consultados de forma a facilitar a entender, por exemplo, a distribuição da variável objetivo em relação aos demais atributos, ou qualquer outra análise que o pesquisador ache necessária. Estratégias para lidar com os problemas de qualidade dos dados encontrados já podem ser definidas.

2.3.3 Preparação dos Dados

Na preparação dos dados, os atributos que realmente serão utilizados para resolver o problema são extraídos no volume que for desejado, caso a disponibilidade de dados seja muito ampla. Também nessa etapa os dados são limpos de acordo com as estratégias definidas anteriormente. Diferentes fontes de dados podem ser mescladas, se necessário, e transformações podem ser feitas para gerar atributos derivados e formatar os dados de acordo com o formato necessário para a utilização de algoritmos de aprendizado de máquina.

2.3.4 Modelagem

Na modelagem, os algoritmos e técnicas a serem utilizados são escolhidos e modelos são gerados com eles. A forma de avaliar o resultado também é definida nessa etapa, pois isso pode levar a tarefas necessárias antes de criar o modelo. Por exemplo, se for decidido que a validação de uma classificação será feita a partir de um conjunto de dados separado, a separação dos dados para treino e validação deverá ser feita antes do modelo ser gerado.

2.3.5 Avaliação

A etapa de avaliação vai analisar os resultados obtidos pelos modelos gerados na etapa anterior. São escolhidas métricas, como por exemplo a acurácia, para comparar os diversos modelos. Também é necessário avaliar se o modelo é capaz de atender ao objetivo do negócio definido na primeira etapa do CRISP-DM. Uma revisão do processo como um todo pode ser feita para avaliar se o objetivo do projeto foi cumprido ou se será necessário repetir alguma etapa.

2.3.6 Implantação

A etapa de implantação representa o fim de um projeto de aprendizado de máquina, mas é essencial para que sua execução traga valor para o negócio. Nela, deve ser definido como será feita a implantação e uso dos modelos gerados. Também é importante planejar o monitoramento e manutenção do modelo, de forma que seu uso siga trazendo resultados relevantes.

Capítulo 3

Trabalhos Relacionados

Nesse capítulo, são apresentados alguns trabalhos já realizados sobre evasão de empregados de empresas. Foram considerados trabalhos com empregados das mais diversas áreas de atuação, uma vez que os oficiais do Exército Brasileiro também têm diferentes especialidades. Durante a pesquisa, percebeu-se que o tema de evasão também é muito utilizado para prever a saída de clientes de empresas, principalmente as de telecomunicações, e que há sobreposição das técnicas utilizadas para tratar esses dois problemas. Dessa forma, alguns estudos relevantes que tratem de evasão de clientes também foram considerados.

3.1 Artigos mais recentes de revisão da literatura

Ahn, Hwang et al. [10] fizeram uma análise das técnicas mais utilizadas para predição de rotatividade nas diversas áreas em que esse tipo de predição é utilizada. Eles mostraram que, atualmente, a maior parte das pesquisas do tipo focam na perda de clientes para as indústrias de jogos de celular e de telecomunicações, mas que também existem em outras áreas, como na rotatividade de empregados em empresas e na indústria de energia. Nos mais de cem artigos analisados pelos pesquisadores, foi identificado que poucos utilizaram técnicas de engenharia de dados para melhorar os atributos utilizados na previsão. Com relação às técnicas utilizadas para a predição, os autores perceberam que elas se encaixavam sempre em quatro grandes áreas: aprendizado de máquina tradicional, estatística, teoria de grafos e *deep learning*. As técnicas de aprendizado de máquina tradicionais foram as mais utilizadas, sendo que os algoritmos mais comuns foram árvores de decisão, regressão logística, support vector machines (SVM), random forest, boosting e k-nearest neighbors (KNN). Os pesquisadores também analisaram as formas mais utilizadas de avaliar o desempenho dos modelos de predição, sendo a área sobre a curva ROC (AUC) a mais comum.

Madane e Chitre [11] fizeram um estudo do mesmo tipo, porém menos extensivo, focando apenas nos algoritmos de predição utilizados na área de predição de rotatividade de empregados e nos resultados obtidos por esses algoritmos. Apenas dez artigos foram analisados, e, entre esses, o algoritmo que teve melhor desempenho no geral foi o random forest, com SVM, KNN, naive Bayes e redes neurais obtendo também bons resultados.

3.2 Estudos de caso sobre predição da saída de funcionários

Entre os artigos publicados nos últimos três anos, vemos que os algoritmos de aprendizado de máquina tradicionais são os mais utilizados. Srivastava e Eachempati [12] compararam os resultados dos métodos *ensemble* random forest e gradient boosting com uma rede neural profunda (*deep learning*), utilizando dados relacionados à carga de trabalho e nível de satisfação do empregado. Os dados utilizados foram obtidos no site Kaggle e eram provenientes de um questionário realizado com cinquenta pessoas. O algoritmo que obteve o melhor resultado foi a rede neural profunda, porém a pequena quantidade de dados utilizados pode ser considerada uma limitação para esse resultado.

Teng, Zhu, Liu e Xiong [13] testaram a influência que os relacionamentos entre empregados têm sobre a decisão de saírem da empresa. Para isso, utilizaram duas técnicas de *deep learning*, graph convolutional networks (GCN) e long short-term memory (LSTM), e dados de redes sociais organizacionais, como a quantidade de e-mail e mensagens trocadas entre os funcionários. Os resultados obtidos foram comparados com algoritmos tradicionais de aprendizado de máquina, resultando em um melhor desempenho nos algoritmos de *deep learning*. Em um trabalho anterior [14], o grupo utilizou redes neurais em dados que indicavam a quantidade de interações anuais entre empregados de uma empresa para tentar prever a saída de empregados por influência dos outros. Novamente, o resultado foi comparado com algoritmos tradicionais e também nesse caso a rede neural que era objetivo do estudo obteve o melhor resultado.

Yuan [15] tentou prever a saída de empregados e entender quais os atributos que mais influenciavam nessa decisão. Foram utilizados dados de mais de dez mil funcionários com 43 atributos relacionados a diversos aspectos de suas vidas pessoal e profissional, como satisfação com o trabalho, oportunidades de crescimento na carreira, gênero, idade, escolaridade, anos na empresa, estado civil, entre outros. Os algoritmos utilizados foram regressão logística, árvores de decisão, redes neurais, SVM e random forest, sendo os dois últimos os que obtiveram um melhor desempenho. Com relação aos atributos que mais influenciaram no resultado estavam chance de ser promovido, comprometimento com a organização, idade e o departamento em que eles trabalham.

Joseph, Udupa et al. [16] trazem uma abordagem interessante para a pesquisa por associar dados pessoais e de satisfação com o trabalho com a possibilidade dos funcionários terem depressão. Todos os dados são obtidos a partir de questionários e são utilizados os algoritmos SVM, árvore de decisão, random forest, naive Bayes, regressão logística e KNN. O melhor resultado foi obtido com o random forest. Além disso, o trabalho detecta que atributos relacionados à vida pessoal do empregado têm uma maior influência do que os atributos relacionados à estrutura fornecida pela empresa.

Jain, Tomar e Jana [17] decidiram abordar o problema de rotatividade de empregados categorizando as pessoas por nível de produtividade na empresa, de forma que o resultado obtido possa também ser melhor utilizado para a criação de políticas de retenção dos funcionários. A ideia dos pesquisadores é que o princípio de Pareto se aplicaria à realidade das empresas e 80% do lucro seria gerado por 20% dos empregados. Nessa abordagem, o problema foi dividido em duas fases. Na primeira, atributos pessoais relacionados à vida na empresa, como quantidade de projetos feitos e a área de trabalho, foram utilizados para agrupar os funcionários. Na segunda, o algoritmo de classificação CatBoost foi utilizado para tentar prever a evasão de funcionários de cada grupo. O CatBoost foi comparado com SVM, regressão logística, árvores de decisão, random forest e extreme gradient boosting, e obteve melhores resultados do que os demais, principalmente para empregados de áreas consideradas críticas na quantidade de pedidos de demissão de funcionários.

Zhao et al. [18] apresentam uma abordagem diferente dos demais por realizar comparações nos resultados encontrados para diferentes tamanhos de conjuntos de dados. Foram utilizados dois conjuntos de dados diferentes, os quais passaram por amostragens para simular dados de empresas pequenas, médias e grandes, mantendo a taxa de evasão apresentada no conjunto de dados original. Ao aplicar os mesmos algoritmos em todas as amostras, o grupo concluiu que o tamanho da amostra é mais relevante do que o conjunto de dados ao determinar o melhor algoritmo. De forma geral, também concluíram que algoritmos baseados em árvores apresentaram os melhores resultados.

Alguns autores também trataram da questão do desbalanceamento das classes nos conjuntos de dados. Gao, Wen e Zhang [19], ao utilizar o algoritmo weighted random forest para compensar esse desbalanceamento, obtiveram resultados melhores do que os de algoritmos tradicionais. Alduayj e Rajpoot [20] compararam o desempenho dos algoritmos SVM, random forest e KNN em três experimentos, sendo o primeiro com o conjunto de dados original e desbalanceado, o segundo com a utilização do algoritmo ADASYN, que tenta balancear os dados criando instâncias sintéticas baseadas na distribuição da classe minoritária, e o terceiro com a realização da subamostragem manual do conjunto de dados. Os melhores resultados foram encontrados no segundo experimento com o algoritmo KNN com $k = 3$.

Uma outra forma de tratar a questão do desbalanceamento das classes é a utilização de métodos de detecção de *outliers*. Essa abordagem, apesar de incomum [10], pode trazer bons resultados. Ullah et al. [21] utilizaram os algoritmos k-means, local outlier factor (LOF) e cluster-based local outlier factors (CBLOF) num problema de detecção de saída de clientes de um banco, com o terceiro algoritmo apresentando um melhor desempenho para o estudo de caso realizado. O trabalho não apresenta, no entanto, uma comparação com outros métodos.

Pesquisas que levam o tempo em consideração para realizar a predição de saída de empregados ainda são minoria na área e nem sempre tentam prever a saída de profissionais no nível individual. Tanto Javed e Azhar [22] quanto Zhu, Seaver, Sawhney et al. [23] tentaram obter uma previsão da quantidade de pessoas a deixar uma empresa, tentando obter um padrão nesse quantitativo a partir de dados consolidados mensalmente.

3.3 Abordagens relacionadas em outras áreas de estudo

Por outro lado, se olharmos para as pesquisas na previsão de saída de clientes de empresas, as quais utilizam técnicas semelhantes às pesquisas de demissão voluntária de empregados [10], estudos mais aprofundados podem ser encontrados. Óskarsdóttir, Van Calster, Baesens et al. [24] utilizaram dados de ligações entre clientes de uma empresa de telecomunicações para montar uma série temporal com o tempo que cada cliente passou, semanalmente, em ligação com clientes da mesma empresa, clientes de outras empresas e com clientes da mesma empresa que deixaram a empresa. Um algoritmo de classificação derivado de random forest, chamado similarity forest, foi utilizado para tentar prever a saída dos clientes para as próximas quatro semanas após o fim das observações. Os resultados foram comparados com os algoritmos clássicos regressão logística e random forest. A conclusão apresentada mostrou que, para as duas primeiras semanas, os algoritmos clássicos tiveram melhor resultado, mas para previsões para quatro semanas o algoritmo similarity forest foi o melhor.

É possível encontrar pesquisas relacionadas ao planejamento de carreiras militares, mas poucos são os trabalhos que focam no problema de evasão. Zais e Zhang [25] trabalharam nesse assunto utilizando dados do Departamento de Defesa dos Estados Unidos da América combinados a dados de média salarial para a área de atuação do militar no mercado civil. Diferentemente da maioria dos estudos de predição de evasão, eles não utilizaram um método de classificação, mas sim um de programação dinâmica, as cadeias de Markov. O método é utilizado para modelar a progressão da carreira militar e estimar a probabilidade do militar permanecer no Exército. No entanto, os autores consideram que essa abordagem

não deve substituir a utilização de algoritmos de classificação, mas sim suplementar, com o modelo criado servindo de entrada para a utilização de outros algoritmos.

3.4 Quadro resumo dos trabalhos relacionados

A tabela 3.1 mostra um resumo dos trabalhos relacionados mencionados nesse capítulo, relacionando os algoritmos testados, o tipo de dado utilizado por cada um deles e o método utilizado para a comparação dos modelos. O algoritmo que obteve o melhor resultado foi destacado em negrito.

Algoritmos	Conjunto de Dados	Método para comparação	Referência
Random forest, gradient boosting, **rede neural profunda**	Satisfação do empregado. Questionário com 365 respostas. Apenas atributos numéricos.	Acurácia	Srivastava e Eachempati (2021)
Rede neural profunda, GCN, LSTM	Quantidade de e-mails e mensagens trocados entre departamentos por mês. 20 atributos, 70 departamentos e 44 meses. Empresa de tecnologia chinesa.	Erro médio absoluto	Teng, Zhu, Liu e Xiong (2021)
Regressão logística, árvores de decisão, redes neurais, **SVM, random forest**	Dados pessoais e profissionais. 43 atributos e 10833 amostras, cada uma correspondendo a um empregado.	Área sobre a curva ROC	Yuan (2021)
SVM, árvores de decisão, **random forest**, naive Bayes, regressão logística, KNN	Dados pessoais, profissionais e dados médicos (depressão), todos provenientes de questionários.	Acurácia	Joseph, Udupa et al. (2021)
CatBoost, SVM, regressão logística, árvores de decisão, random forest, XGB. TOPSIS para categorizar empregados antes da classificação.	Dados pessoais e de produtividade dos funcionários. 10 atributos e 14999 amostras retiradas do site Kaggle.	Coeficiente de correlação de Matthews (MCC)	Jain, Tomar e Jana (2021)
Weighted Random Forest	Dados pessoais e profissionais de empregados de uma empresa de telecomunicações chinesa. 15 atributos e 2000 amostras. Taxa de evasão de 13,5%.	Área sobre a curva ROC, f-score e recall	Gao, Wen, Zhang (2019)
SVM, random forest, **KNN**/ originais, **ADASYN**	Dados sintéticos IBM. 32 atributos, 1470 amostras. 12 atributos selecionados no melhor resultado.	F-score	Alguayj e Rajpoot (2019)
K-means, LOF, **CBLOF**	Dados de uso de conta bancária (customer churn) de um banco Paquistanês.	Precisão, recall e f-score	Ullah et al (2019)
Similarity Forest, regressão logística, random forest	Dados de ligação entre clientes (customer churn)	Área sobre a curva ROC e teste de Kruskal-Wallis	Óskarsdóttir, Van, Calster, Baesens et al. (2018)

Tabela 3.1: Quadro resumo dos trabalhos relacionados.

Percebe-se que redes neurais profundas tendem a obter o melhor desempenho nos estudos de caso em que são utilizadas. No entanto, ainda é mais comum a realização de estudos de caso apenas com algoritmos de classificação mais tradicionais ou com alguma das suas variações. Algoritmos *emsamble* que utilizam árvores, como random forest, CatBoost e similarity forest, aparecem com frequência entre os melhores de seus estudos de caso. Mesmo assim, algoritmos como KNN e SVM também mostram bons desempenhos, indicando que o teste do melhor algoritmo deve ser feito caso a caso e é difícil obter uma generalização para o problema de rotatividade.

Capítulo 4

Fundamentação Teórica

Nesse capítulo, será apresentada a fundamentação teórica dos conceitos mais relevantes para a pesquisa. A escolha das referências foi feita a partir dos principais conceitos encontrados nos trabalhos relacionados encontrados ao aplicar a metodologia TEMAC e também a partir das tecnologias utilizadas no desenvolvimento do estudo de caso que será apresentado no capítulo 5.

A seção 4.1 trata do problema de rotatividade propriamente dito, mostrando como ele foi estudado ao longo do tempo. A seção 4.2 mostra alguns dos métodos de pré-processamento utilizados na preparação de dados para utilização em uma classificação. A seção 4.3 explica algoritmos de classificação utilizados no estudo de caso. A seção 4.4 mostra algoritmos para reamostragem de dados desbalanceados. A seção 4.5 apresenta o problema de detecção de *outliers* e explica alguns dos algoritmos utilizados para resolver esse problema. Por fim, a seção 4.6 discorre sobre testes estatísticos utilizados para comparar modelos de aprendizagem de máquina, com foco no teste de McNemar.

4.1 O problema de rotatividade de empregados

O problema de rotatividade de empregados vem sendo estudado há mais de um século. Desde a década de 1920 [6], pesquisadores das áreas de psicologia e gerência de recursos humanos perceberam os impactos que um pedido de demissão voluntária de um empregado pode causar para uma empresa e, com isso, tentam entender o que leva um funcionário a pedir demissão e prever quem fará esse tipo de pedido, numa tentativa de possibilitar que a empresa possa prevenir ou ao menos diminuir o impacto dessa saída.

Nas últimas duas décadas, essas tentativas de prever quais funcionários deixarão uma empresa passaram a ser feitas no campo da computação com a utilização de algoritmos de aprendizado de máquina [18]. Técnicas de inteligência artificial que se popularizaram inicialmente para resolver problemas das áreas de publicidade e vendas [26], como

por exemplo a rotatividade de clientes de empresas de telecomunicações, passaram a ser utilizadas também na área de recursos humanos, pois, apesar dos dois problemas terem diferenças nas motivações e impactos resultantes da evasão, os mesmos algoritmos apresentam bons resultados para os dois casos [4].

Os motivos identificados para a saída de um profissional não mudaram muito com o passar dos anos. Entre os motivos positivos para a demissão voluntária, estão a oferta de um emprego mais interessante em outra empresa, com melhores salários, condições de trabalho, benefícios ou possibilidades de crescimento na carreira. Já entre os motivos negativos estão os conflitos com colegas de trabalho ou chefes e a falta de trabalhos interessantes, treinamentos e possibilidades de crescimento na carreira [4].

Pode-se encontrar uma relação entre essas motivações e os tipos de dados mais utilizados por estudos dessa predição. Em sua maioria, eles são relacionados a aspectos da vida pessoal e profissional do empregado, como idade, gênero, departamento em que trabalha, distância entre a casa e o trabalho, nível de escolaridade, salário, tempo que já está na empresa, tempo desde a última promoção, entre outros [18, 26]. A maior parte dos estudos utiliza dados de apenas um momento dos empregados e alguns poucos acrescentam o fator temporal, obtendo todos os atributos para todos os empregados em janelas de tempo determinadas [4], podendo assim avaliar a evolução do profissional na empresa. O problema costuma ser tratado com algoritmos de classificação em que a classe a ser prevista é a demissão e normalmente ocorre uma comparação de algoritmos, sendo que entre os mais populares estão árvores de decisão, random forest, suport vector machines (SVM), redes neurais, gradient boosting e k-nearest neighbors (KNN).

Apesar de bons resultados serem encontrados, eles costumam ser específicos para o problema que vão resolver, não sendo possível generalizar qual o melhor método a ser usado [18]. O que se encontra, normalmente, são algumas indicações de quais seriam os melhores algoritmos dependendo do tipo de atributos utilizados, do desbalanceamento entre as classes e do tamanho do conjunto de dados [18], mas mesmo essas indicações podem não corresponder ao encontrado na prática em outros estudos de caso.

4.2 Pré-processamento dos dados

O pré-processamento dos dados é uma etapa fundamental a ser realizada para garantir bons resultados nos modelos de classificação. Considera-se que ao menos 80% do esforço despendido em um projeto de mineração de dados é gasto nessa etapa do projeto [27]. Essa etapa inclui atividades como o tratamento de dados nulos ou faltantes, a normalização dos dados e a engenharia de atributos, na qual se enquadram métodos de seleção de atributos

e a transformação de atributos categóricos em numéricos. A seguir serão explicados alguns métodos de transformação dos atributos categóricos em numéricos e de normalização.

4.2.1 Transformação dos atributos categóricos

A maioria dos algoritmos de aprendizado de máquina só aceitam dados numéricos como entrada. No entanto, diversos conjuntos de dados contém dados categóricos, o que dificulta a utilização direta desses dados no treinamento de modelos de aprendizado de máquina. Para resolver esse empasse, diversas técnicas que realizam a transformação de atributos categóricos em numéricos surgiram [28]. A escolha do método afeta diretamente no resultado dos modelos gerados com os dados, principalmente nos casos em que há atributos de alta cardinalidade presentes e, apesar de existirem indicações de casos em que cada método é indicado, por vezes ainda é necessário realizar testes para encontrar o melhor método para cada caso.

One Hot Enconding

O one hot encoding é um dos métodos mais comumente usados para transformar atributos categóricos nominais, tendo a utilização mais recomendada para atributos com baixa cardinalidade [27] e em conjunto com algoritmos não baseados em árvores.

Nesse método, cada valor possível de um atributo é transformado em um novo atributo de valor binário, tendo o valor "1" atribuído caso ele represente o valor do ponto de dado e recebendo o valor "0" caso contrário. A figura 4.1 mostra um exemplo de como isso ocorre para um atributo com três valores possíveis.

ESTADO_CIVIL
SOLTEIRO
CASADO
SOLTEIRO
DIVORCIADO
CASADO
CASADO

SOLTEIRO	CASADO	DIVORCIADO
1	0	0
0	1	0
1	0	0
0	0	1
0	1	0
0	1	0

Figura 4.1: Exemplo do funcionamento do One Hot Encoding

Target Encoding

O target encoding surgiu como uma alternativa a ser utilizada para atributos de alta cardinalidade, também sendo mais recomendado ao utilizar algoritmos baseados em árvores.

Na versão para classificação binária desse método, considera-se que X são os possíveis valores que um atributo pode ter e $Y \in \{0, 1\}$ são os valores que a variável alvo, ou seja, a classe a ser prevista, pode ter. O método calcula para cada valor X_i que X pode assumir a probabilidade de que $Y = 1$ dado que $X = X_i$ [27]. Isso equivale a calcular a média do valor da variável que representa a classe para cada valor possível do atributo que está sendo codificado e substituir o valor do atributo pelo resultado desse cálculo. A figura 4.2 exemplifica melhor como esse cálculo ocorre.

Figura 4.2: Exemplo do funcionamento do Target Encoding

Os valores resultantes para cada atributo estarão entre 0 e 1, estando normalizados. Esse método deve ser utilizado apenas no conjunto de treinamento de um modelo, pois de outra forma ocorrerá uma situação de vazamento de dados, que tornará a predição feita posteriormente incorreta. Vazamento de dados é o nome dado à situação em que o algoritmo, durante a etapa de treinamento, tem acesso a dados que não estarão disponíveis no momento de fazer a predição, o que acaba influenciando a capacidade de generalização do modelo gerado.

Leave-one-out Encoding

Leave-one-out encoding é um método derivado do target encoding e muito similar a ele. A diferença entre os dois métodos é que, ao calcular a probabilidade que será o novo valor do atributo, o leave-one-out não considera a própria instância para a qual está realizando o cálculo. Isso é feito com a intenção de reduzir a influência de dados anômalos no resultado da codificação [29]. A figura 4.3 mostra um exemplo de como isso acontece. O novo valor para estado civil corresponde à média da classe para cada valor original de estado civil sem considerar a própria linha que está sendo calculada.

4.2.2 Normalização dos atributos numéricos

A normalização dos atributos é uma etapa do pré-processamento dos dados fundamental para a utilização de algoritmos de classificação. Alterar a escala de todos os atributos para um mesmo intervalo de valores é o que impede que atributos com valores altos

ESTADO_CIVIL	CLASSE	NOVO_ESTADO_CIVIL
SOLTEIRO	0	1
CASADO	0	1/3
SOLTEIRO	1	0
CASADO	0	1/3
CASADO	1	0
CASADO	0	1/3

Figura 4.3: Exemplo do funcionamento do Leave-one-out Encoding

sobreponham os atributos com valores mais baixos durante o treinamento dos modelos [30]. Assim como na transformação de atributos categóricos, existem diversos métodos para realizar a normalização e a escolha do mais adequado depende do problema a ser resolvido.

Normalização Z-score

Essa normalização tem como objetivo fazer com que a média de todos os elementos de um atributo tenham valor zero e que o seu desvio padrão seja unitário. Para isso, calcula-se a média dos valores (μ) e o seu desvio padrão (δ). A seguir, a média é subtraída de cada elemento x e o valor resultante é dividido pelo desvio padrão [30], como indicado na fórmula

$$z = \frac{x - \mu}{\delta}$$

Essa normalização também é encontrada na literatura com outros nomes. Na biblioteca *scikit-learn*, que é uma das mais utilizadas bibliotecas em *Python* para a resolução de problemas de aprendizado de máquina, ela é implementada pela classe StandardScaler.

Normalização Min-Max

Essa normalização distribui os valores dos atributos de forma linear em um intervalo pré-determinado, normalmente entre 0 e 1. Para o caso desse intervalo, o novo valor de um elemento de um atributo é calculando subtraindo o valor mínimo que esse atributo possui do valor do elemento e esse resultado é dividido pela diferença entre o valor máximo e o valor mínimo do atributo [30], conforme a fórmula

$$z = \frac{x - min(x)}{max(x) - min(x)}$$

Esse método, em comparação com a normalização Z-score, preserva melhor a relação entre os elementos presentes no dado de entrada. No entanto, ela acaba sendo mais sensível à presença de *outliers*. Na biblioteca *scikit-learn*, ela é implementada pela classe MinMaxScaler.

Robust Scaler

Essa normalização, presente na biblioteca *scikit-learn*, é uma variação da normalização Z-score feita com a intenção de ser mais robusta para a presença de *outliers* no conjunto de dados. Nela, ao invés da média, cada elemento é subtraído da mediana do atributo. Depois, o valor resultante é dividido pela diferença entre o valor do terceiro e do primeiro quartil do atributo [31], conforme a fórmula

$$z = \frac{x - x_{med}}{Q3 - Q1}$$

A imagem 4.4 indica visualmente o que são os quartis. A mediana equivale ao $Q2$.

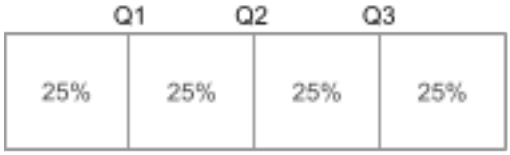


Figura 4.4: Separação dos dados em quartis

4.3 Algoritmos de classificação

A seguir, será apresentado de forma resumida o funcionamento e casos de uso indicados para os principais algoritmos de classificação utilizados em estudos de caso sobre a rotatividade de empregados.

4.3.1 K-Nearest Neighbors (KNN)

No KNN, cada atributo é considerado uma dimensão no espaço R^n. Quando um novo registro deve ser classificado, a distância entre ele e todos os outros registros já existentes é calculada e os k pontos mais próximos são selecionados. A classe do novo registro será a mais comum entre esses k pontos selecionados. Para o cálculo dessa distância,

diversas métricas podem ser utilizadas, sendo que entre as mais comuns estão a distância Euclidiana e a de Manhattan [32].

A figura 4.5 mostra um exemplo do que ocorre no treinamento desse algoritmo numa classificação com três classes possíveis para um valor de $k = 15$ e apenas dois atributos, o que permite que os dados sejam mostrados num espaço bidimensional. Qualquer novo dado encontrará os 15 pontos mais próximos de onde ele está e será classificado de acordo com a classe da maioria desses pontos de dados.

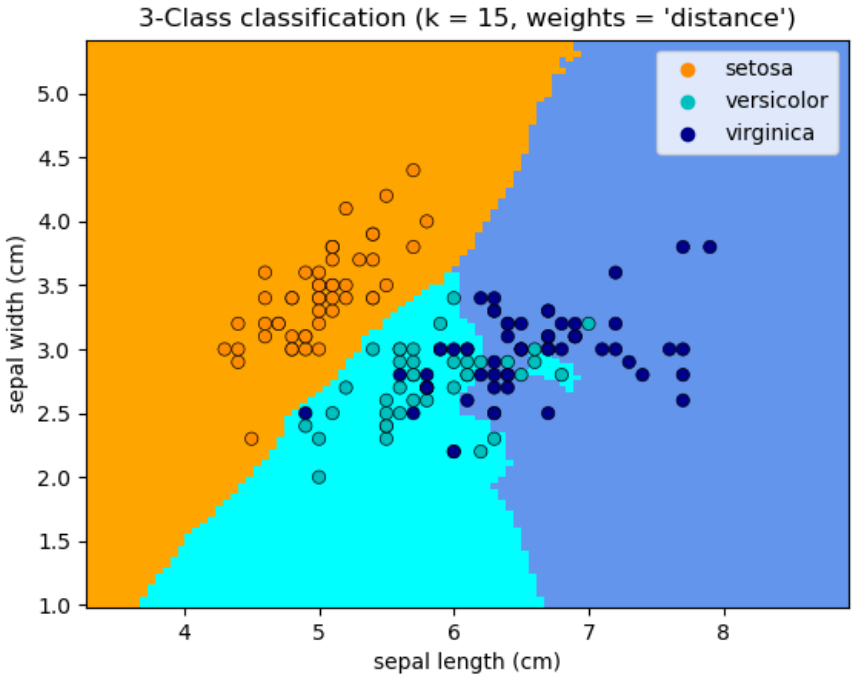


Figura 4.5: Exemplo de classificação com KNN [2]

Para ser utilizado, portanto, é necessário que os atributos tenham valores em que é possível calcular essa distância. Uma limitação desse algoritmo é a diminuição de sua precisão com o aumento do número de atributos utilizados [18]. Entre os estudo de caso relacionados, ele só foi o melhor da comparação feita em [20] com um conjunto de dados sintético com informações de 1470 empregados em 32 atributos, todos eles tendo uma representação numérica.

4.3.2 Árvores de Decisão

Árvore de decisão é o nome dado para os algoritmos que constroem uma estrutura de árvore para realizar a classificação. Nessa estrutura, um novo registro percorre os nós a partir da raiz. Cada nó representa uma decisão sobre um dos atributos que indicará para qual nó filho o registro deve seguir, com a classificação sendo decidida ao atingir um

nó folha. A figura 4.6 mostra um exemplo de classificação com árvore de decisão e três classes, cada uma representada por uma cor. A primeira linha em cada caixa indica o atributo e valor que decidem a divisão. A linha iniciada por *value* apresenta uma lista com a quantidade de elementos de cada classe naquele nó ao construir a árvore.

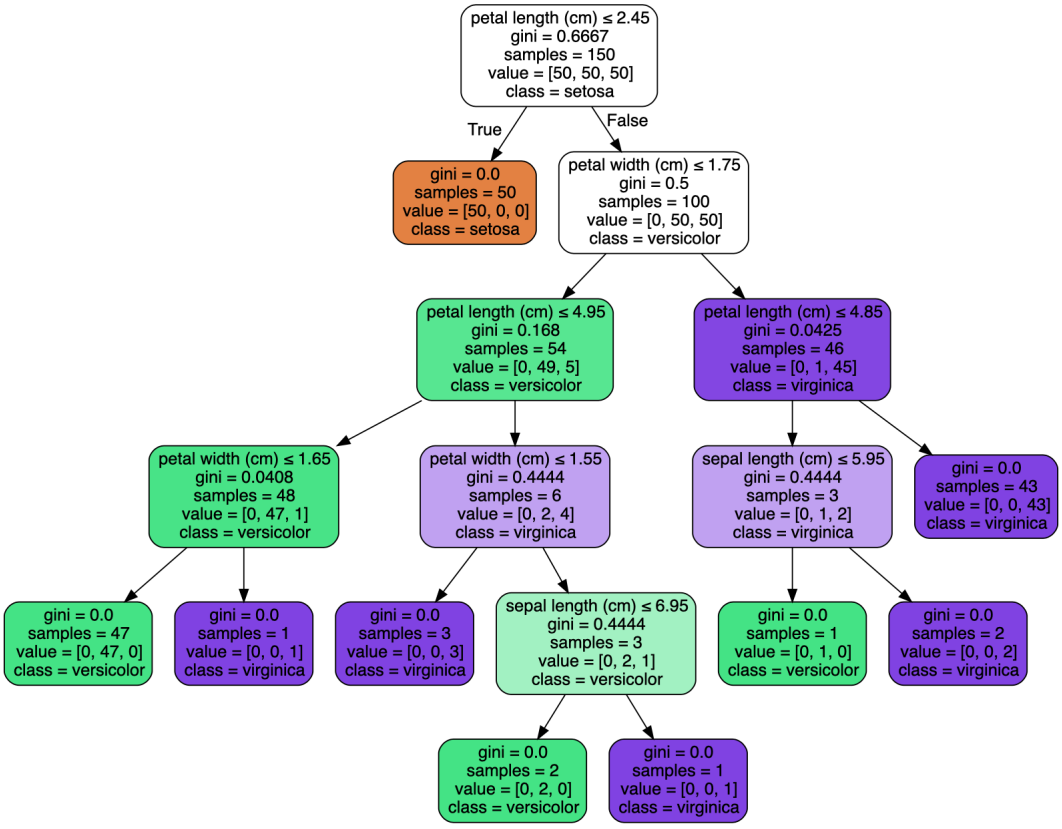


Figura 4.6: Exemplo de classificação com árvore de decisão [3]

A construção da árvore é comumente feita de forma recursiva. Para cada nó é identificado o melhor ponto de separação, ou seja, um atributo e um valor que levarão aos nós seguintes garantindo que esses nós serão os mais puros em relação à classe. O processo é repetido com os nós criados anteriormente até que cada nó folha tenha apenas elementos de uma classe [32] ou até atingir algum outro critério de parada pré-definido, como uma profundidade máxima da árvore ou um número mínimo de elementos no nó. Existem diversas formas de calcular essa pureza dos nós para definir o ponto de separação, sendo duas das mais comuns o índice Gini e o ganho de informação.

O ganho de informação é a métrica utilizada no algoritmo C4.5, que é um dos mais conhecidos para árvores de decisão. Ele é calculado pela diferença entre a entropia antes e depois da divisão do nó, ou seja, a entropia do nó pai menos a entropia dos nós filhos [33]. A entropia, por outro lado, é calculada pela fórmula $E = -\sum_{i=1}^{N} p_i \times \log_2(p_i)$, em que N é o número de classes e p_i é a probabilidade de um elemento naquele nó pertencer à

classe i. Quanto mais puro um nó, ou seja, quanto mais elementos de uma mesma classe ele tiver, menor será a entropia do nó. Escolher uma divisão para um nó que maximize o ganho de informação equivale a escolher o atributo e valor que fará a entropia dos nós filhos diminuírem o máximo possível em relação ao nó pai.

A outra métrica utilizada no lugar do ganho de informação é o índice Gini, calculado pela fórmula $G = 1 - \sum_{i=1}^{N} (p_i)^2$. O índice Gini é interpretado como o cálculo da probabilidade de um elemento ser classificado incorretamente ao ter sua classe escolhida de maneira aleatória [34]. Quanto mais baixo for o valor, mais puro é um nó. Assim, o objetivo ao utilizar essa métrica é escolher o atributo e valor que minimizarão o índice calculado. Uma vantagem desse índice em relação ao uso do ganho de informação é que ele não utiliza a função logarítmica no seu cálculo, tornando mais rápida a execução. O algoritmo de árvore de decisão mais conhecido que faz uso desse índice é chamado de CART.

As duas formas de decidir como dividir o nó da árvore de decisão estão implementadas na biblioteca *scikit-learn* e podem ser escolhidas em um dos parâmetros da classe DecisionTreeClassifier. Esse tipo de classificador é simples de entender conceitualmente, é capaz de lidar com diversos tipos de atributos e apresenta resultados fáceis de interpretar [18], o que faz dele um algoritmo muito utilizado. Apesar de não ter sido o melhor algoritmo em nenhum dos estudos de caso relacionados estudados no capítulo 3, seu estudo ainda é interessante por ser a base dos algoritmos random forest e gradient boosting, esses sim encontrados entre os que apresentaram bom desempenho nos trabalhos relacionados.

4.3.3 Random Forest

Random forest é um algoritmo do tipo *ensamble* que combina diversas árvores de decisão para classificar os dados [18]. Nele, as observações do conjunto da dados passam por amostragens aleatórias com o método *bootstrap*, ou seja, uma amostragem aleatória com reposição, gerando K conjuntos de treino diferentes com o mesmo tamanho do conjunto de dados inicial. Uma árvore de decisão é criada para cada um dos K conjuntos de dados de treino, formando assim a floresta a que o nome do algoritmo se refere. O resultado da classificação se dá por meio de votação, ou seja, será o resultado mais comum encontrado entre as árvores [19]. Além disso, o algoritmo também pode limitar o número de atributos que serão considerados ao fazer uma divisão nas árvores geradas, escolhendo os atributos que serão considerados de maneira aleatória.

A adição dessas duas aleatoriedades na construção das árvores é feita para reduzir a chance de ocorrer *overfitting* e facilitar o cálculo da importância dos atributos para a classificação, além de já ter se mostrado uma técnica capaz de realizar classificações com grande acurácia [35]. De fato, entre os estudos de caso analisados no capítulo 3, esse

algoritmo ou alguma variação dele foi o que mais figurou entre os melhores resultados obtidos.

4.3.4 Gradient Boosting

Gradient boosting é um método ensamble no qual vários modelos mais fracos são combinados de forma iterativa para conseguir um classificador mais forte [36]. Uma das implementações mais comuns que se encontra na literatura utiliza árvores de decisão de tamanho fixo para criar os modelos fracos.

Esse algoritmo tem como objetivo encontrar uma função que mapeie os dados de entrada em valores da saída minimizando uma função de perda escolhida. O algoritmo começa escolhendo um valor constante que miniminiza essa função de perda. A partir daí, vai, iterativamente: calcular o resíduo entre os valores preditos e o valor real da saída; treinar uma árvore tendo o valor desses resíduos como variável alvo; calcular uma nova saída, que será usada na próxima iteração; e ajustar o modelo a partir do resultado encontrado, uma vez que esse algoritmo é do tipo aditivo e utiliza a saída do modelo anterior para encontrar o novo [37]. A etapa iterativa é feita M vezes, sendo M uma quantidade de árvores escolhida como parâmetro do algoritmo.

O gradient boosting atualmente não é tão utilizado em sua forma original, porém segue sendo importante por ser a base de dois algoritmos amplamente utilizados e que costumam apresentar bons resultados ao resolver problemas de classificação, o XGBoost e o CatBoost. Em um dos estudos relacionados analisados no capítulo 3 o CatBoost aparece como sendo o que apresenta o melhor desempenho no seu estudo de caso e o XGBoost costuma estar entre os algoritmos utilizados por ganhadores de competições na plataforma Kaggle.

XGBoost

XGBoost é um algoritmo ensamble de árvores de decisão baseado no gradient boosting e criado com o objetivo de ser altamente escalável. Ele se diferencia do gradient boosting normal pela função de perda utilizada, que é uma função específica capaz de controlar a complexidade das árvores geradas [36]. Isso, entre outros benefícios, resulta em modelos que são treinados mais rapidamente que o gradient boosting comum e utilizando um menor espaço na memória. Ele costuma estar entre os algoritmos que melhor performam em diversas competições de resolução de problemas com aprendizado de máquina.

CatBoost

CatBoost é um algoritmo que foi desenvolvido para resolver um problema de vazamento de dados encontrado no gradient boosting original e na maioria das variações existentes anteriormente chamado *prediction shift* [38]. Para conseguir resolver esse problema, o algoritmo utilizado é modificado em relação ao gradient boosting original, mas ainda é considerado um algoritmo de *boosting* por utilizar vários modelos fracos para construir iterativamente um modelo forte.

Além de resolver esse problema, outra vantagem do CatBoost em relação a outras implementações é sua capacidade de lidar com dados categóricos diretamente, principalmente com atributos que possuem uma grande quantidade de valores possíveis, ou seja, alta cardinalidade. Ele implementa uma transformação própria nos atributos desse tipo que é similar à forma com que os modelos são construídos [36], permitindo que o problema do *prediction shift* não ocorra nessa etapa também.

4.4 Algoritmos de reamostragem para conjuntos de dados desbalanceados

Na resolução de problemas de classificação, é comum encontrar conjuntos de dados desbalanceados, ou seja, em que as duas classes não aparecem no conjunto de dados com proporções parecidas. No geral, a presença de elementos da classe de interesse é muito menor do que a quantidade de elementos da outra classe e, por isso, é de interesse reduzir a quantidade de erros de classificação para essa classe mais rara [39]. A utilização de conjuntos de dados desbalanceados pode ser bem prejudicial para o desempenho de alguns algoritmos de classificação. Dessa forma, métodos de reamostragem foram criados para balancear o conjunto de dados. Esses métodos podem utilizar técnicas de *undersampling*, nas quais a quantidade de elementos da classe majoritária é reduzida, técnicas de *oversampling*, em que dados da classe minoritária são introduzidos no conjunto de dados, ou uma combinação dos dois tipos de técnicas.

4.4.1 SMOTE

O algoritmo SMOTE - *Synthetic Minority Over-sampling TEchnique* - é uma técnica de *oversampling* na qual dados sintéticos da classe minoritária são criados e inseridos no conjunto de dados. Ele se diferencia de outras técnicas de *oversampling* em que os dados da classe minoritária são apenas replicados no conjunto de dados.

Para criar os dados sintéticos, o SMOTE busca os k vizinhos mais próximos da classe minoritária, seleciona aleatoriamente entre esses k vizinhos uma quantidade de dados

que vai depender da quantidade de *oversampling* necessária a se realizar, e então cria dados que ficam entre cada dupla de pontos selecionados. Para a criação do novo dado, o algoritmo vai calcular a diferença no valor de cada atributo dos dados selecionados e multiplicar essa diferença por um valor aleatório entre 0 e 1 [39]. Como resultado, o conjunto de dados final terá regiões de decisão maiores e menos específicas, facilitando a classificação.

Ao longo do tempo, algumas variantes do SMOTE surgiram para tentar melhorar os resultados obtidos com o uso dessa técnica. Duas delas, o borderline SMOTE e o SVM SMOTE, serão descritas a seguir.

Borderline SMOTE

Muitos dos algoritmos de classificação tentam identificar o limite (*borderline*) entre as classes para conseguir classificar os dados corretamente. A partir dessa observação, o borderline SMOTE foi criado com o objetivo de só aumentar a quantidade de amostras de dados da classe minoritária que estão presentes nesse limite entre as classes, tornando assim a separação entre as classes mais clara para os algoritmos de classificação [40].

A definição se um dado está ou não na faixa limite das duas classes vai depender da proporção de dados da classe minoritária e majoritária entre os k vizinhos mais próximos de um dado. Se mais da metade, porém não todos, os vizinhos de um dados da classe minoritária são da classe majoritária, então ele é considerado um dado do limite. A situação em que todos os vizinhos são da classe majoritária é desconsiderada pois, nesse caso, considera-se que o dado da classe minoritária é um ruído e, portanto, não deve ser amplificado. Para gerar os dados sintéticos, a abordagem é similar à do SMOTE original, ou seja, são criados pontos aleatórios entre um dado da classe minoritária e algum de seus k vizinhos mais próximos que também é da classe minoritária.

SMV SMOTE

No SVM SMOTE, assim como no borderline SMOTE, só serão criados dados da classe minoritária na região limite entre as duas classes. No entanto, nessa abordagem, a definição dessa região limite ocorre com a utilização do classificador SVM.

Para esse algoritmo, o classificador é treinado, o limite entre as duas classes é determinado e os pontos da classe minoritária que estão nessa fronteira são identificados. A partir daí, para cada ponto identificado, são observados os seus k vizinhos mais próximos. Se mais da metade deles for da classe majoritária, o algoritmo faz uma interpolação, ou seja, cria um novo ponto entre o ponto observado e cada um dos seus vizinhos da classe minoritária. Por outro lado, se a maioria dos vizinhos for da classe minoritária, o algoritmo faz uma extrapolação. Nesse caso, o novo ponto também é criado na mesma linha

entre o ponto observado e cada um dos seus vizinhos da classe minoritária, porém, ao invés de ser entre os pontos, será fora deles, na direção do lado da classe majoritária na fronteira, expandindo assim essa classe [41].

4.4.2 ADASYN

O algoritmo ADASYN (*Adaptative Synthetic*) é uma técnica de *oversampling* em que os novos dados da classe minoritária são adicionados de forma adaptativa de acordo com sua dificuldade de aprendizagem, ou seja, os dados que são mais difíceis de aprender são multiplicados mais vezes do que os demais.

O algoritmo começa determinando a quantidade de dados a ser gerados, o que vai depender do nível de desbalanceamento inicial do conjunto de dados e do nível de desbalanceamento que se pretende atingir após a aplicação da técnica. A seguir, é determinada a dificuldade de aprendizado de cada dado da classe minoritária. Nessa etapa, o algoritmo encontra os k vizinhos mais próximos de cada dado dessa classe e calcula a razão entre a quantidade de vizinhos que pertencem à classe majoritária em relação ao total k de vizinhos. Os valores encontrados para cada dado da classe minoritária são então normalizados e multiplicados pela quantidade de dados a ser gerados encontrada inicialmente, obtendo assim a quantidade de novos dados a ser adicionada para cada dado da classe minoritária presente no conjunto de dados original. Para gerar esses novos dados, é utilizada uma técnica similar ao SMOTE: para cada dado da classe minoritária, um dos seus k vizinhos, também da classe minoritária, é selecionado aleatoriamente. O novo dado será criado selecionando aleatoriamente um ponto entre esses dois dados [42].

4.4.3 Near Miss

Near Miss é uma técnica de *undersampling* proposta em [43] que apresenta três versões diferentes, NearMiss-1, NearMiss-2 e NearMiss-3. Todas essas versões selecionam quais elementos da classe majoritária vão manter, baseando-se na distância euclidiana desses elementos para seus vizinhos da classe minoritária.

No caso do NearMiss-1, serão mantidos os elementos com a menor distância média para os 3 vizinhos mais próximos da classe minoritária. Já para o NearMiss-2, são escolhidos os elementos com a menor distância média para os três vizinhos mais distantes da classe minoritária. Diferentemente das duas primeiras versões, o NearMiss-3 apresenta duas etapas. Na primeira, os M vizinhos mais próximos de um elemento da classe minoritária são identificados. A seguir, o algoritmo mantém o elemento da classe majoritária pertencente a esse conjunto selecionado que tem a maior distância média para os três vizinhos mais próximos da classe minoritária, assim como o NearMiss-1 [44]. A diferença

que a primeira etapa faz nessa versão é de manter apenas os elementos majoritários mais próximos da fronteira entre as classes, que farão mais diferença na habilidade de classificar os elementos corretamente.

4.4.4 Tomek Link

O algoritmo Tomek Link é uma técnica de *undersampling* em que um Tomek Link é detectado e um dos elementos pertencentes a essa ligação é eliminado. Um Tomek Link é uma dupla de elementos de classes opostas que têm um ao outro como vizinho mais próximo. A definição de qual elemento da ligação será removido do conjunto de dados é feita através de um parâmetro do algoritmo, havendo duas opções possíveis: remover os dois elementos ou remover o elemento da classe majoritária [44]. O Tomek Link é comumente utilizado junto do algoritmo de *oversampling* SMOTE, tendo inclusive uma implementação dessa junção fornecida diretamente na biblioteca *imbalanced-learn* [44], a qual fornece a implementação dos diversos dos algoritmos tratados nessa seção.

4.4.5 Edited Nearest Neighbors

O Edited Nearest Neighbors é um algoritmo de *undersampling* em que o algoritmo k-nearest neighbors para três vizinhos é aplicado para definir quais elementos devem ser removidos. São removidos todos os elementos de uma vizinhança quando todos ou a maioria dos vizinhos não são da mesma classe que o elemento inicial. A definição se será usado o critério de comparar todos os elementos ou a maioria depende do valor de um dos possíveis parâmetros passados para o algoritmo [44]. Assim como o Tomek Link, esse é um algoritmo comumente implementado em conjunto com o SMOTE.

4.5 Métodos de detecção de *outliers*

Outliers ou anomalias é o nome dado a objetos que, num conjunto de dados, se comportam de maneira diferente dos demais. A análise de *outliers* faz uso de diversas técnicas para tentar detectar esses casos excepcionais [45]. A utilização de técnicas específicas se faz necessária, pois os métodos tradicionais de aprendizado de máquina tentam encontrar padrões gerais nos dados e tendem a tratar os *outliers* como ruído nesses padrões, não sendo, assim, boas para detectá-los.

A técnica a ser utilizada deve ser escolhida considerando os dados de entrada e o tipo de *outlier* a ser detectado. Segundo Sangeetha e Geetha [45], os *outliers* podem ser classificados em globais, contextuais ou coletivos. *Outliers* globais são dados que, de forma individual, se distinguem dos demais do conjunto de dados. Já os contextuais são aqueles

que dependem do contexto para serem considerados *outliers*. Por exemplo, para decidir se uma temperatura de 30 ºC é um *outlier* é necessário saber em que época do ano e local ela ocorreu. Se no verão brasileiro, é uma ocorrência comum. Se no inverno canadense, certamente é um *outlier*. Por fim, *outliers* coletivos são aqueles que dependem de uma quantidade de ocorrências do mesmo tipo para serem considerados uma anomalia. Um exemplo disso seriam atrasos em entregas dos correios, que em pequena quantidade são normais e esperadas, mas se ocorrerem com uma grande frequência serão uma anomalia.

Existem diversos métodos para detecção de *outliers* e o método ideal vai variar dependendo de cada aplicação [45]. Algo em comum entre muitos dos métodos utilizados para a detecção dos *outliers* é a utilização de uma medida de similaridade para decidir se um ponto é um *outlier* ou não ao compará-lo com os dados ao seu redor. Esses métodos podem fazer uma comparação da distância entre os pontos, ou seja, do raio ao redor de cada dado, como é o caso do k-means, ou comparar a densidade de cada ponto com a dos vizinhos, como é o caso do local outlier factor (LOF) [21].

4.5.1 Local Outlier Factor (LOF)

O algoritmo local outlier factor não define se os dados são *outliers* de forma binária, mas sim calcula um valor que representa o quanto cada dado é um *outlier* ou não. Esse valor é o chamado *local outlier factor* que dá o nome ao algoritmo [46]. O algoritmo considera que um determinado dado é um *outlier* local se sua densidade for consideravelmente diferente da densidade de seus vizinhos mais próximos. A densidade de um ponto vai depender da distância entre o ponto e o mais distante dos seus k vizinhos mais próximos. O valor que determina se o ponto é um *outlier* ou não será a razão entre a densidade de um ponto e a de seus vizinhos mais próximos. Quanto mais próximo de 1 esse valor for, menor é a chance do ponto ser considerado um *outlier*. Para esse algoritmo, a quantidade de vizinhos analisada costuma ser alta se comparada com a quantidade de vizinhos normalmente consideradas em algoritmos como o KNN. A implementação feita pelo *scikit-learn* considera um valor de 20 vizinhos analisados como sendo ideal, podendo aumentar para conjuntos de dados com presença de *outliers* de mais de 10% [47].

4.5.2 Isolation Forest

Isolation forest é um algoritmo que utiliza a construção de árvores para detectar anomalias em um conjunto de dados. Nesse algoritmo, o termo *isolation* significa "separar uma instância das demais", ou seja, isolar um ponto de dado dos demais. A ideia é criar árvores com separações aleatórias e de forma recursiva até o momento em que cada instância esteja em um nó folha. Ao fazer isso, os dados anômalos serão fáceis de identificar, pois terão

uma altura média bem menor do que os demais dados. Isso ocorre pois dados que se destacam dos demais, como é o caso das anomalias, tem maior chance de serem isolados dos demais mais cedo no algoritmo [48].

O algoritmo funciona em duas etapas, a de treinamento e a de avaliação. Na primeira, diversas árvores são criadas até que suas folhas apresentem dados isolados ou até que ela atinja uma profundidade máxima. Na segunda etapa, as árvores são avaliadas e uma nota de anomalia é calculada para cada instância baseada na altura da folha em que a instância ficou. As notas de anomalia de cada árvore para cada instância são combinadas, obtendo uma nota final para cada instância. Essas notas são ordenadas de forma decrescente e a quantidade de anomalias pode ser escolhida a partir dessa lista [48].

4.6 Teste de McNemar para comparação de modelos de aprendizado de máquina

Ao obter modelos a partir da utilização de algoritmos de aprendizado de máquina para resolver um problema, duas perguntas costumam ser feitas na análise do resultado: qual dos modelos obtidos é o melhor e se o modelo encontrado é de fato capaz de resolver o problema, ou seja, se a capacidade preditiva é melhor do que uma adivinhação aleatória. A melhor forma de responder essas perguntas é através do uso de testes estatísticos, uma vez que eles conseguem definir se a diferença nas métricas encontradas em cada modelo são resultado de aleatoriedade ou não.

Um teste estatístico utilizado nessa situação terá uma hipótese nula em que os dois classificadores apresentam a mesma proporção de erros. Se o teste rejeitar essa hipótese, então é possível dizer que os dois classificadores tiveram desempenhos diferentes para o conjunto de validação utilizado. No entanto, esses testes podem apresentar dois tipos de erros. O erro tipo I é a probabilidade do teste rejeitar a hipótese nula incorretamente, enquanto o erro tipo II é a probabilidade do teste não rejeitar a hipótese nula incorretamente [49]. Dessa forma, é importante escolher o teste adequado para cada caso de forma a minimizar esses erros.

O principal critério a considerar nessa escolha é a quantidade de dados disponível para treinamento, teste e validação. Se a quantidade de dados for grande o suficiente para esses conjuntos de dados serem diferentes, sem a necessidade de algum tipo de reamostragem, o teste de McNemar é o teste estatístico que apresenta o menor erro do tipo I [50].

Para a aplicação do teste de McNemar, o conjunto de dados é dividido em dois, um conjunto de treino e um de teste. Dois algoritmos, A e B, são treinados no conjunto de treino e os classificadores obtidos são testados no conjunto de teste. A partir do resul-

tado dessa predição do conjunto de testes, uma tabela de contingência é criada conforme mostrado na tabela 4.1.

Quantidade de elementos classificados incorretamente por A e B (a)	Quantidade de elementos classificados incorretamente por A, mas não por B (b)
Quantidade de elementos classificados incorretamente por B, mas não por A (c)	Quantidade de elementos classificados corretamente por A e B (d)

Tabela 4.1: Tabela de contingência a ser utilizada para o teste de McNemar

O teste de McNemar é calculado com

$$\chi^2 = \frac{(b-c)^2}{(b+c)}$$

e χ^2 segue uma distribuição qui-quadrada com um grau de liberdade. A partir dessa distribuição, é possível perceber que, se a hipótese nula for verdadeira, ou seja, se os classificadores forem equivalentes, a probabilidade de χ^2 ser maior que 3,841 é de menos de 5%. Dessa forma, se o valor de χ^2 for maior do que 3,841, a hipótese nula pode ser rejeitada e o desempenho dos classificadores podem ser consideradas diferentes para o conjunto de dados utilizado [50].

Capítulo 5

Estudo de Caso

O estudo de caso desse trabalho teve por objetivo testar técnicas de aprendizado de máquina para obter modelos de predição da evasão de oficiais do Exército Brasileiro. A seguir, serão descritas as atividades realizadas em cada uma das seis fases do *framework* CRISP-DM, utilizado para a execução desse estudo de caso.

5.1 Entendimento do Negócio

Para entender a questão da evasão de oficiais no Exército Brasileiro, é necessário compreender quem são os militares do EB e como funcionam as diversas carreiras que podem ser seguidas pelos oficiais, pois essas pessoas não formam um grupo homogêneo.

Existem diversas especializações no Exército, as quais definem as formas de ingresso, as funções que poderão ser assumidas e o plano de carreira dos militares. A identificação dessa especialização ocorre por meio da Arma, Quadro ou Serviço que o militar pertence. As Armas englobam as especializações dos militares combatentes, ou seja, aqueles que exercerão a atividade-fim da profissão. Quadros, por sua vez, têm militares com origens mais amplas, sendo que cada quadro tem uma finalidade específica. Nos Serviços estão militares que executam alguma atividade de apoio, normalmente de logística [51].

Diversos aspectos da carreira de um militar dependem da Arma, Quadro ou Serviço a que ele pertence, como o tempo entre promoções, as organizações militares em que ele pode servir, o posto máximo que pode ser alcançado e os cursos que podem ser feitos. Esses aspectos da carreira militar em sua maioria são definidos pelo Departamento-Geral do Pessoal (DGP) e pelas organizações militares subordinadas a ele, como a Diretoria de Avaliação e Promoções (DAPROM), que cuida da promoção de militares, e a Diretoria de Controle de Efetivos e Movimentações (DCEM), que cuida da seleção de militares para vagas nas organizações militares, da movimentação desses militares e da designação de militares para cursos. Como visto nos capítulos 3 e 4, esse tipo de característica da carreira

é considerado um fator relevante na motivação para o pedido de demissão de empregados [4]. Assim, é esperado que militares de diferentes Armas, Quadros ou Serviços apresentem padrões diferentes de evasão.

Por esse motivo, nas etapas seguintes do estudo de caso foram obtidos modelos de previsão considerando todos os oficiais e também separando por carreira. São objeto de estudo dessa pesquisa militares pertencentes às Armas de Infantaria, Cavalaria, Artilharia, Comunicações e Engenharia, ao Quadro de Engenheiros Militares (QEM), ao Quadro Complementar de Oficiais (QCO), ao Quadro de Material Bélico (QMB) e aos Serviços de Intendência e Saúde. Ficarão de fora o Quadro Auxiliar de Oficiais (QAO), por se tratar de um quadro acessado apenas no fim da carreira de praças, que normalmente não saem por pedido de demissão e sim por aposentadoria (reserva), e também o Serviço de Assistência Religiosa, por ter poucas pessoas.

5.2 Entendimento dos Dados

Os dados utilizados foram coletados da base corporativa do Exército Brasileiro, a qual armazena as informações de todos os militares da ativa e da reserva e históricos relativos à carreira dessas pessoas. Ao todo, foram coletados dados de 19.781 militares, dos quais 1.541 pediram demissão. Os dados foram exportados de um banco de dados Oracle para o formato .csv. Esses dados representam a situação dos oficiais na data da coleta e são compostos principalmente de dados pessoais, como sexo, cidade natal, religião, nível de escolaridade, estado civil e quantidade de dependentes, e dados relacionados à carreira, como o posto atual, o Quadro, Arma ou Serviço que o militar pertence, a cidade em que serve atualmente, o curso de formação que fez, data da última promoção e movimentação, ano de formação, entre outros.

Explorando os dados obtidos, pode-se obter uma visão melhor da atual situação de evasão de oficiais no Exército Brasileiro. Analisando a evasão por turma de formação na figura 5.1, percebe-se que a taxa de evasão teve uma tendência crescente dos anos 90 à metade dos anos 2000, seguida de uma diminuição nos anos mais recentes. O pico no meio dos anos 2000 coincide com os primeiros anos após uma grande alteração nas leis que regulam as carreiras militares, o que pode ter deixado alguns oficiais descontentes com a carreira e levado a uma maior evasão.

Ao mudar a análise para a quantidade de demissionários por tempo de serviço, é possível ver que a maioria dos pedidos de demissão ocorrem nos primeiros anos de formado. A figura 5.2 também mostra o posto dos militares que pedem demissão. Apesar dos postos e o tempo de serviço estarem diretamente relacionados pela própria natureza da progressão da carreira militar, essa análise é interessante por permitir notar que a maioria dos pedidos

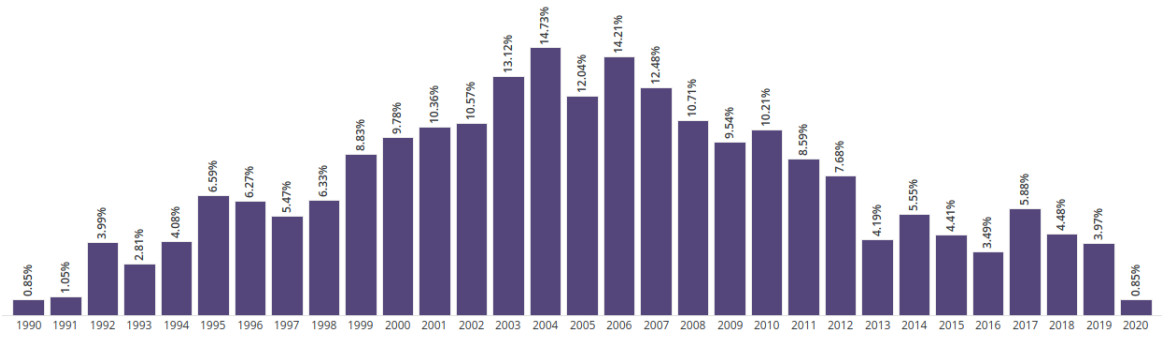


Figura 5.1: Evasão de oficiais do Exército por turma de formação.

de demissão ocorre até o posto de Capitão, tornando-se mais raros após a promoção a Major. Também percebe-se com esse gráfico que a evasão ocorre com maior intensidade até dez anos após a formação, com picos nos anos zero e cinco desde a formação, o que mostra que a tendência decrescente vista nas turmas da última década na figura 5.1 não pode ser realmente considerada uma tendência, pois esses militares ainda estão no tempo de maior risco de evasão. Os picos nos anos zero e cinco da figura 5.2 podem ser explicados por ter militares que saem logo após a formatura, já tendo certeza que não desejam seguir uma carreira militar, e pelos militares que aguardam o fim do prazo de cobrança da multa por sair antes de cinco anos. Possivelmente, nos próximos anos, esse pico nos cinco anos será reduzido enquanto um novo pico deverá surgir nos três anos, uma vez que uma nova alteração da regulamentação das carreiras militares alterou o tempo de duração dessa multa para três anos.

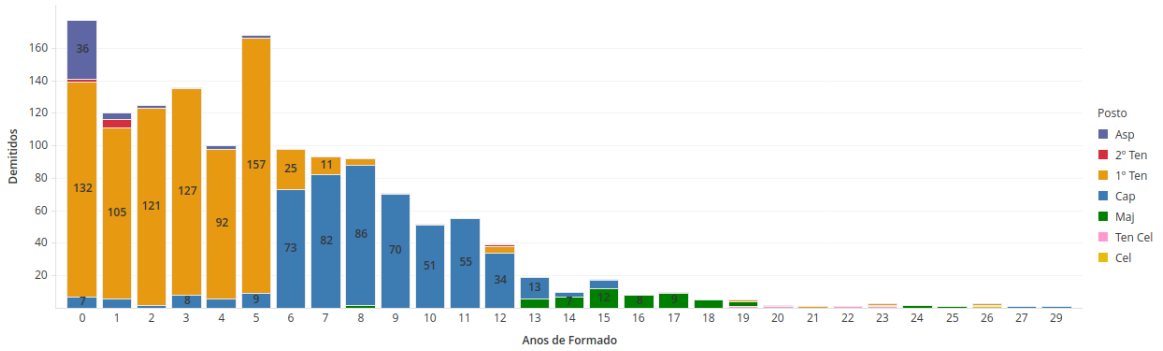


Figura 5.2: Evasão de oficiais do Exército por tempo de serviço e posto

Também é de interesse dos gestores dos recursos humanos do Exército entender a evasão considerando as diferentes carreiras militares, pois as mesmas influenciam em diversos aspectos, como o tempo mínimo para promoção ao próximo posto e nas oportunidades de cursos e funções. Analisando, na figura 5.3, a taxa de evasão por Quadro, Arma ou Serviço, percebe-se que a diferença entre elas é considerável, com uma evasão que ultra-

passa 30% para militares do Quadro de Engenheiros Militares (QEM). Em um distante segundo lugar estão os militares Médicos do Serviço de Saúde, com 13.75% de evasão.

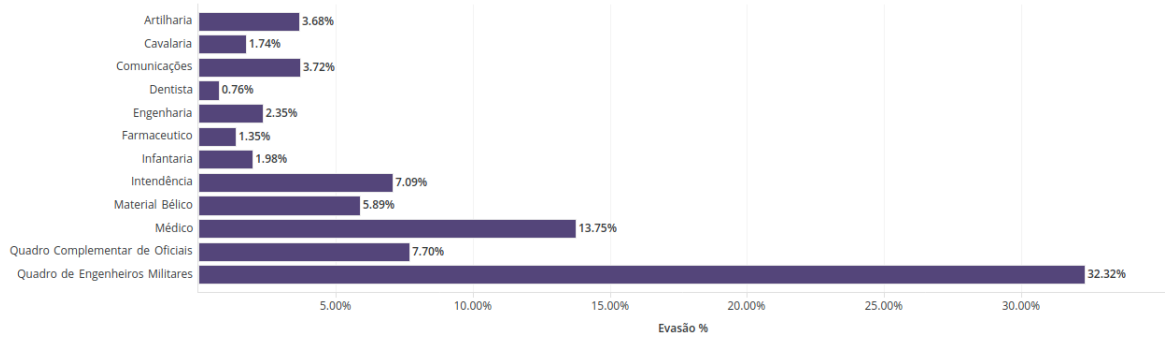


Figura 5.3: Evasão de oficiais do Exército por Quadro, Arma ou Serviço

5.3 Preparação dos Dados

5.3.1 Obtenção do conjunto de dados

Dentre os diversos eventos da carreira militar armazenados em banco de dados, os seguintes foram selecionados para a utilização na predição:

- a formação, com a classificação do militar na turma e a carreira escolhida;
- a promoção e posto atual;
- a realização de cursos;
- a movimentação para outras organizações militares e a cidade em que elas se encontram;
- o recebimento de medalhas;
- a ida em missões no exterior;
- o cadastro de proficiência em idiomas;
- a ocupação de apartamento funcional, os quais recebem o nome de Próprio Nacional Residencial (PNR);
- a utilização de Licença para Tratar de Interesse Particular (LTIP); e
- o cadastro de dependentes.

Todos esses eventos são armazenados no banco de dados a cada vez que acontecem e contêm a data em que ocorreram entre os dados registrados. Além disso, os seguintes

dados pessoais foram selecionados: sexo, religião, estado civil, nível de escolaridade e cidade natal.

Foi decidido que os dados utilizados para a predição mostrariam a situação de cada militar no dia da extração dos dados, tornando necessário realizar um agrupamento dos eventos mencionados acima para permitir que cada pessoa seja representada por apenas uma linha no conjunto de dados, independentemente da quantidade de eventos que ocorreu ao longo da sua carreira. Esse agrupamento dos eventos foi feito através da contagem da quantidade de vezes que cada um deles ocorreu e da data da última ocorrência. Numa primeira tentativa de utilizar os dados para predição, percebeu-se que a data da última ocorrência dos eventos serviria de indicativo que um militar já pediu demissão há algum tempo. Isso leva a um problema chamado vazamento de dados, em que o modelo tem acesso a uma informação que não estaria disponível da mesma forma em casos novos, ou seja, militares prestes a pedir demissão.

Por exemplo, considere um militar que pediu demissão no posto de primeiro-tenente, cinco anos após sua promoção. Se a coleta dos dados for realizada dez anos após seu pedido de demissão, esse militar terá uma data de última promoção de quinze anos antes dessa coleta. Essa quantidade de tempo estará acima da de qualquer outro primeiro-tenente que ainda esteja na ativa, uma vez que o tempo máximo nesse posto é de dez anos, dependendo do Quadro, Arma ou Serviço a que o militar pertence. Com isso, será fácil para o algoritmo de classificação identificar um padrão que o destaca dos demais e torna claro que ele pediu demissão.

Dessa forma, foi necessário trocar os atributos de data pelo tempo em anos passado desde a última ocorrência do evento. Para os militares que se demitiram, esse tempo foi calculado subtraindo a data do evento da data de demissão. Já para os militares que ainda estão na ativa, foi calculada a diferença da data do evento para a data de extração dos dados. No exemplo do primeiro-tenente que pediu demissão, ao realizar o cálculo do tempo desde a última promoção conforme esse método, esse militar terá o tempo de cinco anos, não sendo possível diferenciá-lo dos demais militares da ativa que realmente foram promovidos há cinco anos do momento da coleta dos dados.

Por fim, decidiu-se que nem todos os eventos precisam ter o quantitativo e o tempo da última ocorrência no conjunto de dados utilizado. Para a ocupação de PNR e a utilização de LTIP, os dois atributos foram trocados por um boleano que indica se essa é ou não a situação atual do militar ou se era no momento do pedido de demissão. Como se trata de eventos mais raros na carreira de um militar, a maioria das pessoas não teve uma ocorrência desses tipos e eles têm um impacto mais pontual na vida do militar, não sendo tão relevante a sua ocorrência no passado. Para o cadastro de habilitação em idiomas, ida a missões no exterior, recebimento de medalhas, realização de cursos

e cadastro de dependentes, foram mantidos apenas o quantitativo de ocorrências, pois são eventos que podem ocorrer mais de uma vez e cuja influência na carreira ou vida pessoal do militar é a mesma a partir da sua ocorrência, não importando tanto o tempo passado desde a ocorrência deles. Já para promoção, foi mantido apenas o tempo da última ocorrência, uma vez que o posto atual também é um atributo e o formato da carreira torna o quantitativo para esse atributo uma informação redundante. A formação também só ficou com o tempo da última ocorrência, pois ela só acontece uma vez para cada militar. O único atributo que ficou tanto com o quantitativo quanto com o tempo da última ocorrência foi a movimentação, sendo que o tempo ainda foi dividido em tempo na Organização Militar (OM) e tempo na guarnição. O primeiro indica o tempo que o militar não troca de local de trabalho e o segundo o tempo que o militar não troca de cidade de trabalho.

Muitos dos dados extraídos são representados no banco de dados por um código e uma descrição do atributo. Para a predição, apenas um deles será necessário, por serem representações diferentes da mesma informação. De forma geral, os dados que se encontram nessa situação são atributos categóricos nominais e, portanto, serão mantidas as descrições, de forma a facilitar o entendimento dos dados nas manipulações futuras. As duas exceções são posto e nível de escolaridade, que são atributos categóricos ordinais. Nesse caso, será mantido o código, pois ele mantém a ordem entre os diferentes valores do atributo, que é uma informação adicional em relação à simples descrição do atributo.

A obtenção dos dados foi finalizada com a avaliação da qualidade do dados e filtragem de dados defeituosos. Os valores possíveis de cada atributo foram avaliados separadamente e, para os atributos com valores estranhos ou nulos, a quantidade de militares com a anomalia foi considerada para definir o que fazer com aquele valor. Foram encontrados valores nulos para a Organização Militar (OM), a classificação na turma e o estado civil e valores negativos para a quantidade de dias em missão no exterior, para o tempo desde o fim do curso e para o tempo desde a última promoção. Além disso, resolveu-se retirar os oficiais generais e os coronéis do conjunto de dados, por se tratar de um grupo com características muito diferentes dos demais militares e que nunca tem demissionário, uma vez que todos têm tempo para pedir a aposentadoria (reserva) remunerada ou estão muito próximos disso.

Ao todo, 390 registros se enquadravam em alguma das situações mencionadas acima, o que representa 2,0% do total. Por essa porcentagem ser baixa e causar pouca alteração na taxa de evasão total, foi decidido pelo descarte desses dados. A alteração que causou maior redução na quantidade de dados restantes foi a remoção dos postos de general e coronel. Com todas as alterações de preparação dos dados finalizadas, restaram 15.540 registros para utilização, dos quais 1.435 pediram demissão. A taxa de evasão dos dados a

serem utilizados é de 9,23%. Outros dois problemas encontrados na qualidade dos dados ocorreram com o tempo que o militar está na mesma OM ou guarnição e com o nível de escolaridade. Para os dois casos, havia muitos militares com valores nulos, de forma que a remoção deles representaria uma perda considerável para o conjunto de dados e alteraria a taxa de evasão geral. Dessa forma, foi decidido desconsiderar esses dois atributos no conjunto de dados utilizado.

A tabela 5.1 lista o nome dos 24 atributos selecionados após essas preparações e da variável a ser prevista. Também fornece uma pequena descrição do que cada atributo representa e de seu domínio e indica se são dados categóricos nominais, categóricos ordinais ou numéricos.

5.3.2 Pré-processamento dos dados

Entre a obtenção dos dados e a sua efetiva utilização na modelagem, por vezes é necessário realizar algumas etapas de pré-processamento de dados, que os preparam para o formato esperado por cada tipo de algoritmo a ser utilizado. Essas etapas são consideradas um pré-requisito para utilizar alguns algoritmos, mas também são recomendadas para melhorar os resultados obtidos pelos estimadores utilizados. Para cada tipo de operação de pré-processamento existem diversas técnicas disponíveis e as escolhas feitas nessa etapa têm grande potencial de influenciar o resultado do modelo obtido posteriormente com os algoritmos de classificação. Por isso, os classificadores testados nesse trabalho foram combinados com algumas operações de pré-processamento. Foram testadas diversas combinações de pré-processadores e classificadores e os resultados obtidos foram analisados para identificar qual combinação apresenta melhor capacidade de prever os militares demissionários.

Um dos tipos de operações de pré-processamento de dados comumente realizado é o tratamento de valores faltantes. No conjunto de dados dos demissionários do Exército, os valores faltantes foram tratados juntamente com a extração dos dados, removendo linhas que tinham valores nulos ou removendo atributos que apresentavam valores majoritariamente vazios conforme já explicado. Assim, esse tipo de operação não será feita novamente. Devido ao tipo de dados dos atributos escolhidos, duas técnicas foram usadas para pré-processar os dados: transformação dos atributos categóricos em numéricos e normalização dos atributos numéricos. As técnicas foram aplicadas respectivamente nos atributos categóricos nominais e nos atributos numéricos. Não foi necessário aplicar as técnicas de normalização nos atributos categóricos codificados, pois as técnicas de transformação utilizadas já trazem resultados normalizados.

Para a implementação das técnicas de pré-processamento e posteriormente na implementação dos classificadores, foi utilizada majoritariamente a biblioteca *Python scikit-*

COLUNA	DESCRIÇÃO	TIPO_DADO
POSTO	Representação numérica do posto do militar. Sequência decrescente em que 11 representa Aspirante e 6 representa Tenente-coronel.	categórico ordinal
TEMPO_SERV_ANOS	Tempo em anos desde a entrada no exército. Inteiro entre 0 e 42.	numérico
QAS_QMS	Especialidade do militar, combinando qualificação e sub-especialidade. 121 valores possíveis.	categórico nominal
QUALIF_DESCRICAO	Descrição da qualificação (especialidade geral do oficial). 12 valores possíveis.	categórico nominal
SEXO	Booleano em que 0 representa Masculino e 1 representa Feminino.	categórico nominal
CURSO	Nome do curso de formação feito pelo oficial. 16 valores possíveis.	categórico nominal
TEMPO_FIM_CURSO_ANOS	Tempo em anos desde a finalização do curso de formação. Inteiro entre 0 e 31.	numérico
OM_SIGLA	Sigla da Organização Militar que o oficial está trabalhando atualmente. 788 valores possíveis.	categórico nominal
CIDADE_NATURAL	Código que representa a cidade natal do oficial. 1029 valores possíveis.	categórico nominal
CIDADE_OM	Código que representa a cidade em que a OM atual do militar está. 156 valores possíveis.	categórico nominal
ESTADO_CIVIL	Código que representa o estado civil do militar. 7 valores possíveis.	categórico nominal
RELIGIAO	Nome da religião do militar. 43 valores possíveis.	categórico nominal
CLASSIFICACAO_TURMA	Valor que representa a classificação final do oficial em sua turma de formação. Inteiro entre 1 e 296.	numérico
QTD_CURSOS	Quantidade que representa a quantidade de cursos realizados pelo militar, sem contar o curso de formação. Inteiro entre 0 e 109.	numérico
QTD_DEPENDENTES	Quantidade que representa a quantidade de dependentes cadastrados pelo militar. Dependentes costumam ser os familiares diretos do militar, como cônjuge e filhos. Inteiro entre 0 e 8.	numérico
QTD_DEPENDENTE_MILITAR	Quantidade dos dependentes que também são militares em atividade. Inteiro entre 0 e 2.	numérico
QTD_MOV	Quantidade de movimentações realizadas ao longo da carreira. Uma movimentação é uma mudança de organização militar. Inteiro entre 0 e 19.	numérico
TEVE_LTIP	Booleano em que 0 representa um militar que não está em Licença para Tratamento de Interesse Próprio e que 1 representa um militar que está utilizando essa licença.	categórico nominal
QTD_MEDALHAS	Quantidade de medalhas recebidas ao longo da carreira. Inteiro entre 0 e 80.	numérico
DIAS_MISSAO_EXTERIOR	Quantidade de dias passados no exterior em alguma missão. Inteiro entre 0 e 2579.	numérico
QTD_IDIOMAS_HABILITADO	Quantidade de idiomas em que o militar tem proficiência, comprovada por prova específica. Inteiro entre 0 e 4.	numérico
OCUPA_PNR	Booleano em que 0 representa um militar que não ocupa uma residência funcional e 1 representa um militar que ocupa uma residência funcional.	categórico nominal
TEMPO_ULTIMA_PROMOCAO	Tempo em anos desde que o militar foi promovido pela última vez. Inteiro entre 0 e 27.	numérico
DEMITIDO	Booleano em que 0 representa o militar que não pediu demissão e 1 representa o militar que pediu demissão voluntária.	booleano

Tabela 5.1: Atributos selecionados com sua descrição e seus tipos de dados.

learn, uma vez que ela já contém a implementação dos principais algoritmos utilizados nesse trabalho e que fornece o recurso de *pipeline*, o qual facilita o uso combinado e sequencial de técnicas para criar modelos. Nos casos em que algum algoritmo utilizado não estiver disponível nessa biblioteca, será dita qual a biblioteca de origem dele quando ele for mencionado nas próximas etapas do CRISP-DM.

Codificação de atributos categóricos

A codificação de atributos categóricos foi aplicada nos atributos destacados na tabela 5.2.

COLUNA	TIPO_DADO
POSTO	categórico ordinal
TEMPO_SERV_ANOS	numérico
QAS_QMS	categórico nominal
QUALIF_DESCRICAO	categórico nominal
SEXO	categórico nominal
CURSO	categórico nominal
TEMPO_FIM_CURSO_ANOS	numérico
OM_SIGLA	categórico nominal
CIDADE_NATURAL	categórico nominal
CIDADE_OM	categórico nominal
ESTADO_CIVIL	categórico nominal
RELIGIAO	categórico nominal
CLASSIFICACAO_TURMA	numérico
QTD_CURSOS	numérico
QTD_DEPENDENTES	numérico
QTD_DEPENDENTE_MILITAR	numérico
QTD_MOV	numérico
TEVE_LTIP	categórico nominal
QTD_MEDALHAS	numérico
DIAS_MISSAO_EXTERIOR	numérico
QTD_IDIOMAS_HABILITADO	numérico
OCUPA_PNR	categórico nominal
TEMPO_ULTIMA_PROMOCAO	numérico
DEMITIDO	booleano

Tabela 5.2: Atributos selecionados para passar pela codificação.

Os atributos categóricos nominais que não foram marcados possuem apenas dois valores possíveis e já estão codificados com os valores 0 e 1, enquanto o único atributo categórico ordinal existente também já tem uma representação numérica. Por esse motivo, esses atributos foram tratados como atributos numéricos nessa etapa de pré-processamento e foram normalizados.

Como explicado no capítulo 4, a escolha do método de codificação depende de algumas características dos atributos a serem codificados e também do tipo de algoritmo de classificação a ser usado posteriormente. Após a análise dos atributos, que levou em conta suas cardinalidades, a aceptabilidade da perda de informações com a codificação e os algoritmos de classificação a serem usados depois, foi possível optar por três tipos diferentes de codificadores para testagem: One Hot Encoding, Leave One Out Encoding e Target Encoding. Como apenas o primeiro deles estava implementado no *scikit-learn*, foi necessário encontrar uma outra biblioteca que trouxesse a implementação dos outros dois. A biblioteca escolhida foi a *category_encoders* [52], presente na lista de contribuições

externas do *scikit-learn*, uma vez que ela já é adaptada para ser utilizada com as demais funcionalidades do *scikit-learn*.

Cada um desses codificadores apresenta um conjunto de parâmetros que podem ser definidos em sua utilização. A tabela 5.3 mostra os parâmetros e a respectiva lista de valores testados nos codificadores selecionados. Dois dos parâmetros mencionados para o One Hot Encoding têm apenas um valor mencionado. Esse valor foi mencionado na tabela por se tratar de algo diferente do padrão do codificador, mas que precisou ter um valor único para o correto funcionamento dele com o conjunto de dados desse estudo de caso.

	parâmetro	**valores testados**
OneHotEncoder	handle_unknown	ignore
	sparse	TRUE
	drop	None, first
	min_frequency	None, 0.05, 0.1, 0.25
	max_categories	None, 5, 10, 50
LeaveOneOutEncoder	sigma	None, 0.05, 0.1, 0.25, 0.6
TargetEncoder	smoothing	0.0, 1.0, 10.0

Tabela 5.3: Parâmetros testados em cada algoritmo de codificação.

Normalização dos atributos numéricos

Os atributos que não passaram pela etapa de codificação foram submetidos à etapa de normalização. Assim como na etapa anterior, foram levadas em consideração as características dos dados para escolher os métodos de normalização a serem testados. Aqui, foi analisado se os dados formavam uma matriz esparsa, a possibilidade de haver *outliers* e a existência de valores poucos comuns em alguns atributos. Decidiu-se por testar os seguintes normalizadores: normalização z-score, representada pela classe StandardScaler da biblioteca *scikit-learn*, normalização min-max, representada pela classe MinMaxScaler, e o Robust Scaler.

Diferentemente dos codificadores, os normalizadores foram todos utilizados com seus parâmetros padrão. Isso foi decidido após alguns testes mostrarem que a variação nos parâmetros dos normalizadores pouco alterava o resultado dos modelos gerados.

5.4 Modelagem

Na etapa da modelagem, diversos algoritmos foram testados com o objetivo de encontrar aquele capaz de gerar o melhor modelo para resolver o problema de evasão de oficiais. Independentemente do algoritmo testado, a forma de realizar esse teste foi essencialmente a mesma e consistiu nos seguintes passos:

1. os parâmetros do algoritmo foram identificados e diferentes valores de teste foram definidos para cada um deles.
2. o algoritmo foi colocado em uma *pipeline* juntamente com as etapas de pré-processamento apresentadas na seção anterior. *Pipeline* é um recurso do *scikit-learn* que permite a concatenação de vários dos métodos disponíveis na biblioteca para facilitar a implementação dessas técnicas em sequência.
3. a *pipeline* resultante e a lista de valores de hiperparâmetros a serem testados foram passadas para a classe RandomSearchCV. Essa classe implementa os algoritmos da pipeline com algumas combinações dos hiperparâmetros recebidos de forma a identificar qual combinação resulta no modelo com a maior acurácia. As combinações de hiperparâmetros a serem testadas são selecionadas de forma aleatória. Nesse método, ainda é utilizada a técnica de *cross-validation*.
4. o melhor modelo encontrado pelo RandomSearchCV é utilizado para prever os classes de um conjunto de dados de validação, o qual não foi utilizado no teste. As métricas utilizadas para avaliar o modelo são extraídas do resultado dessa predição.

Além dos hiperparâmetros do classificador, os hiperparâmetros que foram definidos para os codificadores de atributos categóricos, mostrados na tabela 5.3, também foram passados para o RandomSearchCV testar. Todas as combinações de codificador, normalizador e classificador foram testadas e o resultado deles foi comparado para ver qual é capaz de obter o melhor modelo. É interessante ressaltar que algumas combinações de algoritmos e hiperparâmetros testados são incompatíveis e resultam em erro, principalmente quando o codificador de atributos categóricos utilizado é o One Hot Encoder, pois ele transforma os dados em uma matriz esparsa e isso não é aceito quando alguns hiperparâmetros dos classificadores são selecionados. No entanto, foi mais simples tentar todas as combinações e ignorar as que resultavam nesse tipo de erro do que eliminar as combinações erradas antes dos testes.

Os dados colocados na *pipeline* foram divididos em dados de treino/teste e validação em uma proporção de 80% para treino e teste e 20% para validação. Os dados de treino e teste foram divididos utilizando *cross-validation* de três *folds*, o qual foi implementado pela

classe RandomSearchCV. Todas essas divisões de dados foram feitas de forma a manter a proporção de demitidos nos conjuntos de dados separados igual à proporção do conjunto de dados completo. Optou-se por ter um conjunto de dados de validação para garantir que o modelo encontrado pelo RandomSearchCV ainda teria bom desempenho quando aplicado em um conjunto de dados que nunca viu antes. A primeira divisão dos dados foi feita antes de qualquer outra etapa do processamento para garantir que os dados usados nessa validação realmente são novos para o modelo gerado e não ocorreria vazamento de dados.

Uma vez definido o método a ser usado para a testagem de algoritmos, seguiu-se para a definição e execução dos algoritmos a serem testados. Foram realizados três grupos de teste, com diferentes algoritmos em cada grupo. Cada grupo de teste foi executado duas vezes, sendo a primeira vez com o conjunto de dados completo, separado apenas em teste e validação, e a segunda vez com o conjunto de dados separado pelas distintas carreiras militares. A seguir, serão listados os algoritmos testados em cada grupo de teste e os hiperparâmetros que foram escolhidos para testagem com cada um deles. Na seção seguinte, correspondente à etapa de Avaliação do CRISP-DM, serão discutidos os resultados encontrados em cada grupo de teste tanto na execução com o conjunto de dados completo quanto na execução dos conjuntos de dados separados por carreira.

5.4.1 Primeiro grupo de testes

No primeiro grupo de testes, foram utilizados os algoritmos de classificação mais encontrados na literatura relacionada, focando especialmente naqueles que obtiveram melhores resultados nesses outros estudos. Os algoritmos testados foram: KNN, árvores de decisão, random forests, gradient boosting, extreme gradient boosting (XGBoost) e CatBoost. Todos os algoritmos foram aplicados na versão disponibilizada no *scikit-learn*, exceto pelos dois últimos, pois não estão disponíveis nessa biblioteca. Para o CatBoost e para o XGBoost foram usadas duas bibliotecas próprias, a *catboost* e a *xgboost*, respectivamente.

A tabela 5.4 mostra a relação de algoritmos usados, com o nome da classe utilizada na implementação, e os hiperparâmetros testados para cada um deles. A escolha dos hiperparâmetros foi feita de forma que se testassem valores variados dentre os valores possíveis para cada um deles e seguindo as orientações fornecidas pela documentação de cada classe. O CatBoost não teve nenhuma escolha de hiperparâmetros porque, segundo a documentação da implementação do algoritmo que está sendo utilizada, ele não precisa de uma boa escolha de hiperparâmetros para apresentar seu melhor resultado.

	parâmetro	valores testados
KNeighborsClassifier	n_neighbors	1, 2, 3, 4, 5
	weights	uniform, distance
	leaf_size	1, 10, 30, 50, 100
	p	1, 2, 3
DecisionTreeClassifier	max_depth	3, 10, 100
	criterion	gini, entropy
	splitter	best, random
	min_samples_split	2, 5, 10
	max_features	None, sqrt, log2
	min_impurity_decrease	0.0, 0.1, 0.5, 1
RandomForestClassifier	max_depth	3, 10, 100
	criterion	gini, entropy
	min_samples_split	2, 5, 10
	max_features	None, sqrt, log2
	min_impurity_decrease	0.0, 0.1, 0.5, 1
GradientBoostingClassifier	criterion	squared_error, friedman_mse
	min_samples_split	10, 25, 50
	learning_rate	0.01, 0.1, 0.5
	loss	log_loss, exponential
	n_estimators	50, 100, 250
	n_iter_no_change	None, 1, 5, 10
	min_samples_split	2, 5, 10
	max_features	None, sqrt, log2
	min_impurity_decrease	0.0, 0.1, 0.5, 1
XGBClassifier	learning_rate	0.01, 0.1, 0.5
	gamma	0, 1, 5, 10
	subsample	0.5, 0.75, 1
	lambda	1, 2, 10
	alpha	0, 1, 5, 10
	scale_pos_weight	1, 10
	num_parallel_tree	1, 3, 5
CatBoostClassifier	-	-

Tabela 5.4: Parâmetros testados para cada classificador.

5.4.2 Segundo grupo de testes

O segundo grupo de testes da modelagem consistiu na utilização de algoritmos para tratar o desbalanceamento da classe "*sim*" em relação à classe "*não*" através da reamostragem. A abordagem utilizada consistiu em testar uma combinação de duas técnicas de reamostragem, sendo uma de *undersampling* e uma de *oversampling*. As combinações foram feitas tanto manualmente quanto pela utilização de uma classe que já faz a combinação dos dois tipos de técnicas. Os algoritmos de *oversampling* SMOTE, Borderline SMOTE, SVM SMOTE e ADASYN foram testados em conjunto com o algoritmo de *undersampling* Near Miss. Além disso, foram utilizados os algoritmos de técnicas combinadas SMOTE Tomek, que combina SMOTE com Tomek Link, e SMOTEEN, que combina SMOTE com Edited Nearest Neighbors. Uma vez que a biblioteca *scikit-learn* não apresentava a implementação de todos esses algoritmos, decidiu-se por utilizar a implementação da biblioteca *imblearn* nos algoritmos de reamostragem.

Os algoritmos de reamostragem foram acrescentados na *pipeline* criada no primeiro grupo de testes entre as etapas de pré-processamento dos dados e o classificador. Novamente, alguns valores de hiperparâmetros foram definidos para serem testados nesses algoritmos. Todas as combinações possíveis entre as etapas foram testadas. A tabela 5.5 mostra os hiperparâmetros testados para cada algoritmo de reamostragem.

	parâmetro	valores testados
SMOTE	k_neighbors	1, 2, 3, 4, 5
BorderlineSMOTE	k_neighbors	1, 2, 3, 4, 5
	m_neighbors	3, 5, 10, 20
	kind	borderline-1, borderline-2
SVMSMOTE	k_neighbors	1, 2, 3, 4, 5
	m_neighbors	3, 5, 10, 20
	out_step	0.5, 0.1, 1.0
ADASYN	n_neighbors	1, 2, 3, 4, 5
NearMiss	version	2, 3
	n_neighbors	2, 3, 4
	n_neighbors_ver3	2, 3, 4
SMOTETomek	SMOTE	k_neighbors = 1, 2, 3, 4, 5
SMOTEENN	ENN	n_neighbors = 2, 3, 4
		kind_sel = all, mode
	SMOTE	k_neighbors = 1, 2, 3, 4, 5

Tabela 5.5: Parâmetros testados para cada algoritmo de reamostragem.

5.4.3 Terceiro grupo de testes

O terceiro grupo de testes foi o teste de algoritmos de detecção de *outliers* ao invés de um classificador tradicional. Foram testados o isolation forest e o local outlier factor. Novamente, eles foram combinados com as etapas de pré-processamento para obter a combinação que resultaria no melhor modelo. A tabela 5.6 mostra os hiperparâmetros definidos para cada algoritmo.

	parâmetro	valores testados
IsolationForest	n_estimators	50, 100, 250
	max_samples	auto, 0.25, 0.5, 0.75
	contamination	auto, 0.01, 0.1, 0.25, 0.5
	max_features	0.5, 0.75, 1.0
	bootstrap	True, False
LocalOutlierFactor	n_neighbors	10, 20, 50, 100
	leaf_size	10, 30, 50
	p	1, 2, 3
	contamination	auto, 0.01, 0.1, 0.25, 0.5

Tabela 5.6: Parâmetros testados para cada algoritmo de detecção de outliers.

5.5 Avaliação

Na etapa de avaliação do CRISP-DM, os modelos encontrados são analisados para determinar se eles são úteis para resolver o problema proposto ou não. No caso do problema da demissão de oficiais do Exército Brasileiro, como não havia um modelo anterior sendo utilizado, o que vai caracterizar se o modelo é útil ou não é a sua capacidade de prever a saída de pessoas melhor do que uma tentativa de adivinhação baseada na classe mais comum do conjunto de dados, ou seja, se a acurácia do modelo for superior à que se obteria em um algoritmo que sempre classifica com a classe mais comum. Um algoritmo que classifica dessa forma está implementado na biblioteca *scikit-learn* e se chama Dummy Classifier. Dessa forma, a primeira etapa da avaliação dos modelos foi executar esse classificador para o conjunto de dados completo e para o conjunto de dados separado por carreiras, obtendo assim um parâmetro para comparação dos modelos obtidos na etapa anterior.

Na tabela 5.7 estão os valores de acurácia obtidos pelo Dummy Classifier em cada conjunto de dados, além da quantidade de elementos de cada classe presente nesses conjuntos de dados. Para esse classificador, foi utilizado o conjunto de dados completo, antes

da separação em treino/teste e validação. Como na separação a proporção das classes foi mantida, o valor da acurácia calculado dessa forma ou no conjunto separado deve ser aproximadamente o mesmo. A separação em carreira mostrada na tabela foi feita de duas formas: a primeira, considerando o valor do atributo QUALIF_DESCRICAO, presente no conjunto de dados; e a segunda considerando a totalização das qualificações que representam a mesma forma de ingresso no Exército Brasileiro, ou seja, o QCO, as carreiras da área de saúde, as carreiras combatentes e o QEM.

	Acurácia Dummy Classifier	nº elementos classe "sim"	nº elementos classe "não"
Conjunto de Dados Completo	90,77%	1435	14105
QCO	91,87%	172	1943
Médico	83,24%	271	1346
Farmacêutico	98,25%	4	224
Dentista	99,10%	3	329
Total - Saúde	87,23%	278	1899
Infantaria	97,37%	87	3227
Intendência	90,89%	122	1217
Cavalaria	97,72%	30	1284
Engenharia	97,08%	28	931
Artilharia	95,12%	72	1402
Material Bélico	92,25%	54	643
Comunicações	95,43%	34	710
Total - Combatentes	95,66%	427	9414
QEM	60,34%	558	849

Tabela 5.7: Acurácia do DummyClassifier e quantidade de elementos em cada classe pra o conjunto de dados completo e por carreira.

Tendo o resultado desse classificador como parâmetro, foi possível analisar o resultado dos modelos encontrados na etapa anterior. No entanto, para definir o melhor modelo, não foi levado em conta apenas a acurácia encontrada. O objetivo principal do modelo é conseguir detectar pessoas que vão sair. É melhor obter um modelo que identifica incorretamente pessoas que vão sair, ou seja, falsos positivos, do que um modelo que identifica mal as pessoas que realmente saíram, ou seja, apresenta muitos falsos negativos. Uma métrica que é capaz de medir isso é o *recall*, calculado com a quantidade de verdadeiros positivos dividida pela soma de verdadeiros positivos e falsos negativos. Quanto maior o valor dessa métrica, menor a taxa de falsos negativos será em relação à de verdadeiros

positivos. Assim, entre modelos com acurácias próximas, foi dada prioridade àqueles que apresentaram uma melhor medida de *recall* para a classe "*sim*".

Para comparar modelos diferentes, foi utilizado o teste estatístico de McNemar, que permite afirmar se a diferença entre dois modelos é considerável ou se é resultado apenas de aleatoriedade. O teste foi calculado comparando o resultado de uma dupla de algoritmos e eles são considerados estatisticamente diferentes se o *p-value* resultante for menor do que 0,05. Esse teste foi escolhido por ser o teste estatístico que apresenta menor erro quando o conjunto de dados é grande o suficiente para ser possível separar o conjunto de dados de treino e de validação.

A seguir, os resultados dos modelos serão apresentados na ordem em que foram obtidos, ou seja, primeiro os resultados para o conjunto de dados completo, depois os resultados separados por carreira e seguindo os três grupos de teste realizados.

5.5.1 Conjunto de dados completo

Primeiro conjunto de testes

O primeiro conjunto de testes executado no conjunto de dados completo foi o dos algoritmos de classificação junto com as etapas de pré-processamento. A tabela 5.8 mostra os valores de acurácia, precisão, *recall* e *f-score* na melhor combinação de etapas de pré-processamento e parâmetros para cada classificador, enquanto a tabela 5.9 mostra qual foi a combinação de classificador, codificador e normalizador e seus respectivos hiperparâmetros que resultou nas métricas mostradas. Alguns testes executados de maneira prévia já haviam mostrado que era possível obter um *recall* de mais de 60% nessa classificação. Dessa forma, a testagem desconsiderou as combinações que resultaram em modelos com um *recall* menor do que esse.

Classificador	classe	acurácia	precisão	recall	f-score
KNN	não	0,96	0,96	0,99	0,98
	sim		0,91	0,64	0,75
Árvores de Decisão	não	0,96	0,98	0,98	0,98
	sim		0,77	0,76	0,76
Random Forest	não	0,97	0,97	1,00	0,98
	sim		0,95	0,74	0,83
Gradient Boosting	não	0,97	0,98	0,99	0,99
	sim		0,91	0,78	0,84
Extreme Gradient Boosting	não	0,98	0,98	0,99	0,99
	sim		0,93	0,81	0,86
CatBoost	não	0,98	0,98	0,99	0,99
	sim		0,93	0,80	0,86

Tabela 5.8: Métricas calculadas para cada classificador

Classificador	hiperparâmetros	Encoder	hiperparâmetros	Scaler
KNN	weights=uniform p=1 n_neighbors=5 leaf_size=100	LeaveOneOut	sigma=None	StandardScaler
Árvores de Decisão	splitter=random min_samples_split=5 min_impurity_decrease=0.0 max_features=None max_depth=100 criterion=gini	OneHotEncoder	min_frequency=0.05 max_categories=5 drop=first	StandardScaler
Random Forest	min_samples_split=5 min_impurity_decrease=0.0 max_features=sqrt max_depth=None criterion=entropy	OneHotEncoder	min_frequency=0.05 max_categories=50 drop=first	MinMaxScaler
Gradient Boosting	n_iter_no_change=10 n_estimators=250 min_samples_split=5 min_impurity_decrease=0.0 max_features=sqrt loss=log_loss learning_rate=0.1 criterion=squared_error	OneHotEncoder	min_frequency=0.1 max_categories=50 drop=None	StandardScaler
Extreme Gradient Boosting	subsample=0.75 scale_pos_weight=1 num_parallel_tree=5 learning_rate=0.5 lambda=10 gamma=1 alpha=0	OneHotEncoder	min_frequency=0.05 max_categories=50 drop=first	RobustScaler
CatBoost	-	OneHotEncoder	min_frequency=0.05 max_categories=None drop=None	StandardScaler

Tabela 5.9: Melhores parâmetros e pré-processamento para cada classificador

Pelos resultados, todos os algoritmos apresentaram acurácia acima da apresentada pelo Dummy Classifier. Assim como ocorreu em muitos dos trabalhos relacionados, os algoritmos ensamble CatBoost e XGBoost foram os que apresentaram melhores resultados de acurácia e, principalmente, *recall*.

Após a execução do teste de McNemar comparando os algoritmos, foi possível confirmar que todos os algoritmos mostrados são de fato estatisticamente diferentes do Dummy Classifier, mostrando que o resultado melhor não é consequência de aleatoriedade. Ao comparar os resultados entre os algoritmos, foi possível ver que o resultado do XGBoost e do CatBoost são estatisticamente equivalentes. O resultado do random forest e do gradient boosting também foram equivalentes entre si.

Segundo conjunto de testes

Após a execução do primeiro conjunto de testes e a obtenção de dois resultados que conseguiram uma acurácia de 98% e um *recall* de 80%, os demais testes passaram a ter uma nova meta: encontrar uma combinação que resultasse em um modelo melhor do que

os obtidos com CatBoost e XGBoost no primeiro conjunto de testes. Por esse motivo, a decisão de desconsiderar modelos que obtivessem *recall* de menos de 60%, que foi aplicada no primeiro conjunto de testes, foi alterada para desconsiderar modelos com menos de 70% de *recall*.

A tabela 5.10 mostra as métricas do melhor modelo encontrado para cada classificador com a utilização das técnicas de reamostragem. A tabela 5.11 mostra qual foi a combinação de algoritmos e hiperparâmetros que levou a esse resultado. O algoritmo KNN nesse conjunto de testes não teve nenhum resultado com o *recall* superior a 70% e, por isso, foi deixado de fora das tabelas de resultados. Assim como no primeiro conjunto de testes, diversas combinações de algoritmos levaram a resultados similares, principalmente para os classificadores XGBoost e CatBoost, e o resultado mostrado na tabela foi o da combinação com aquele resultado que foi executada primeiro.

Classificador	classe	acurácia	precisão	recall	f-score
Árvores de Decisão	não	0,96	0,97	0,98	0,98
	sim		0,79	0,75	0,77
Random Forest	não	0,96	0.97	0,98	0,98
	sim		0,82	0,75	0,78
Gradient Boosting	não	0,97	0,98	0,99	0,98
	sim		0,87	0,81	0,84
Extreme Gradient Boosting	não	0,97	0,98	0,99	0,99
	sim		0,90	0,80	0,85
CatBoost	não	0,97	0,98	0,99	0,98
	sim		0,89	0,80	0,84

Tabela 5.10: Métricas calculadas para cada classificador com a utilização de um método de reamostragem

Os melhores resultados de cada algoritmo foram comparados ao melhor resultado sem a aplicação de técnicas de reamostragem através do teste de McNemar. Para árvores de decisão e gradient boosting, os resultados foram considerados similares aos encontrados no primeiro conjunto de testes. Para os outros, o primeiro conjunto de testes realmente mostrou um resultado melhor, além de serem mais simples de treinar, já que a adição de uma técnica de reamostragem aumenta o tempo de treinamento pro modelo. Dessa forma, para o conjunto de dados testado, os modelos obtidos com XGBoost e CatBoost no primeiro conjunto de testes ainda são os melhores.

Terceiro conjunto de testes

O terceiro conjunto de testes tentou identificar se as pessoas que pediram demissão poderiam ser tratadas como *outliers* do conjunto de dados e, portanto, poderiam ser identificadas através de um algoritmo próprio para detectar esse tipo de caso. Os resultados

Classificador	hiperparâmetros	Encoder	hiperparâmetros	Scaler	Reamostragem	hiperparâmetros
Árvores de Decisão	splitter=best min_samples_split=10 min_impurity_decrease=0.0 max_features=None max_depth=100 criterion=entropy	OneHotEncoder	min_frequency=None max_categories=None drop=None	StandardScaler	SVMSMOTE	k_neighbors=5 m_neighbors=10 out_step=0.5
					NearMiss	version=1 n_neighbors=3
Random Forest	min_samples_split=2 min_impurity_decrease=0.0 max_features=None max_depth=None criterion=entropy	OneHotEncoder	min_frequency=None max_categories=None drop=None	StandardScaler	SMOTETomek	SMOTE/ k_neighbors=1
Gradient Boosting	n_iter_no_change=None n_estimators=250 min_samples_split=2 min_impurity_decrease=0.0 max_features=None loss=exponential learning_rate=0.5 criterion=squared_error	OneHotEncoder	min_frequency=None max_categories=None drop=None	RobustScaler	SMOTETomek	SMOTE/ k_neighbors=5
Extreme Gradient Boosting	subsample=0.75 scale_pos_weight=1 num_parallel_tree=1 learning_rate=0.5 lambda=10 gamma=0 alpha=1	OneHotEncoder	min_frequency=None max_categories=None drop=None	StandardScaler	SMOTETomek	SMOTE/ k_neighbors=5
CatBoost	-	OneHotEncoder	min_frequency=None max_categories=None drop=None	MinMaxScaler	SMOTETomek	SMOTE/ k_neighbors=5

Tabela 5.11: Melhores parâmetros e pré-processamento para cada classificador com a utilização de um método de reamostragem

apresentados na tabela 5.12 mostram que nenhum dos dois algoritmos testados conseguiu resultados com boa acurácia para o conjunto de dados. A tabela 5.13 lista os hiperparâmetros que geraram esse resultado.

Nota-se, pelo resultado, que esses algoritmos estão classificando muitos dos dados como sendo da classe "sim", o que leva ao *recall* alto para a classe "sim"e à precisão alta para a classe "não". Ainda que um alto *recall* seja desejável para o estudo de caso presente, ainda é preferível um algoritmo que seja capaz de classificar melhor as duas classes.

Classificador	classe	acurácia	precisão	recall	f-score
Isolation Forest	não	0,10	1,00	0,01	0,03
	sim		0,09	1,00	0,17
Local Outlier Factor	não	0,12	0,97	0,04	0,07
	sim		0,09	0,99	0,17

Tabela 5.12: Métricas calculadas para cada algoritmo de detecção de *outliers*

A tentativa de realizar o teste de algoritmos de detecção de *outliers* para a resolução desse problema partiu da interpretação de que o pedido de demissão voluntária poderia ser interpretado como uma anomalia do conjunto de dados. No entanto, esse resultado mostra que esse não é o caso. Como os resultados dos outros conjuntos de teste demonstraram, é possível que algoritmos de classificação encontrem padrões para identificar as pessoas que pedem demissão voluntariamente. Dessa forma, uma outra interpretação possível para *outliers* nesse conjunto de dados seriam pessoas que não estão nesse padrão.

Classificador	hiperparâmetros	Encoder	hiperparâmetros	Scaler
Isolation Forest	n_estimators=250 max_samples=auto max_features=0.75 contamination=0.01 bootstrap=True	OneHotEncoder	min_frequency=0.25 max_categories=10 drop=first	MinMaxScaler
Local Outlier Factor	p=3 n_neighbors=100 leaf_size=50 contamination=auto	OneHotEncoder	min_frequency=None max_categories=5 drop=None	MinMaxScaler

Tabela 5.13: Melhores parâmetros e pré-processamentos para cada algoritmo de detecção de *outliers*

Isso representaria pessoas que têm o padrão de que pedirão demissão, mas não pedem e também as pessoas que têm o padrão de que permanecerão na carreira militar, mas pedem demissão.

Avaliação da importância dos atributos

Após a execução dos testes para o conjunto de dados completo e a avaliação dos resultados, foi possível encontrar dois algoritmos que trouxeram resultados bons em prever a saída de oficiais do Exército, mesmo para um conjunto de dados desbalanceado: o XGBoost e o CatBoost. A tabela 5.14 mostra quais foram as combinações de algoritmos e hiperparâmetros que resultaram nos melhores modelos, bem como a acurácia encontrada em cada um deles.

Experimento	Classificador	hiperparâmetros	Encoder	hiperparâmetros	Scaler	Reamostragem	hiperparâmetros	Acurácia
Primeiro Conjunto de Testes	Extreme Gradient Boosting	subsample=0.75 scale_pos_weight=1 num_parallel_tree=5 learning_rate=0.5 lambda=10 gamma=1 alpha=0	One Hot Encoder	min_frequency=0.05 max_categories=50 drop=first	Robust Scaler	-	-	0,98
	CatBoost	-	One Hot Encoder	min_frequency=0.05 max_categories=None drop=None	Standard Scaler	-	-	0,98
Segundo Conjunto de Testes	Extreme Gradient Boosting	subsample=0.75 scale_pos_weight=1 num_parallel_tree=1 learning_rate=0.5 lambda=10 gamma=0 alpha=1	One Hot Encoder	min_frequency=None max_categories=None drop=None	Standard Scaler	SMOTETomek	SMOTE/ k_neighbors=5	0,97
	CatBoost	-	One Hot Encoder	min_frequency=None max_categories=None drop=None	Min Max Scaler	SMOTETomek	SMOTE/ k_neighbors=5	0,97

Tabela 5.14: Resumo dos resultados apresentados pelos melhores modelos encontrados

O CatBoost, que é um algoritmo baseado em árvores, é otimizado para tratar de dados categóricos sem ser necessário realizar a codificação dos atributos categóricos como foi feito nos conjuntos de teste, pois ele tem uma maneira própria de realizar essa codificação. Dessa forma, decidiu-se criar um modelo apenas desse algoritmo utilizando essa

capacidade, ou seja, sem a utilização de etapas de pré-processamento, para comparar com os dois melhores modelos encontrados no primeiro conjunto de testes.

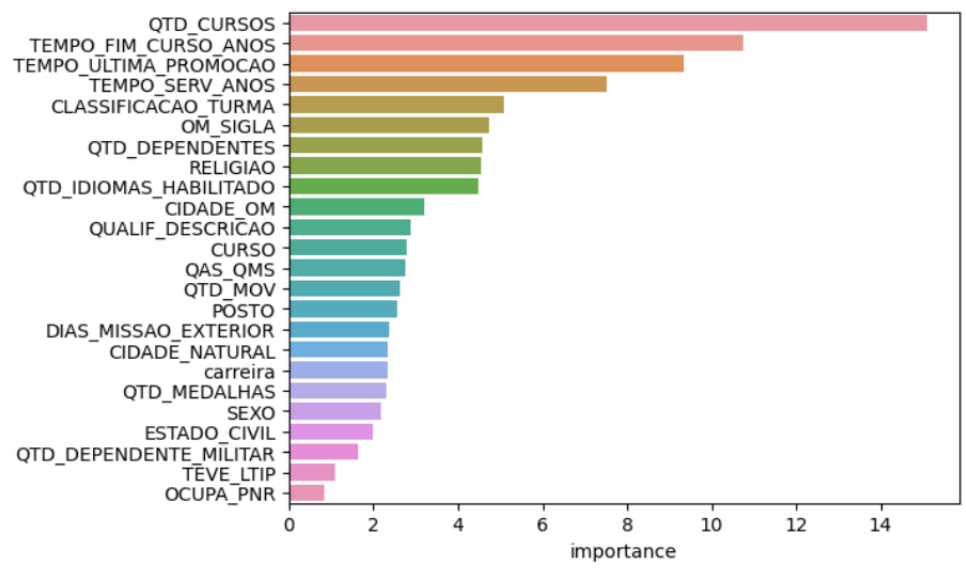
O resultado encontrado com o CatBoost sem pré-processamento foi estatisticamente igual ao do CatBoost do primeiro conjunto de testes, também apresentando acurácia de 98%. No entanto, essa abordagem tem duas vantagens em relação ao primeiro conjunto de testes: o treinamento fica mais rápido, já que o pré-processamento não é necessário, e é mais fácil obter a importância dos atributos para a classificação do que ao utilizar o CatBoost com o One Hot Encoder, uma vez que os atributos não permanecem os mesmos após essa transformação.

A biblioteca *catboost* oferece algumas maneiras de calcular a importância dos atributos. A figura 5.4 mostra dois tipos de importância calculadas para os atributos pelo modelo gerado com o CatBoost sem pré-processamento. A primeira é do tipo 'PredictionValueChange' e é uma medida de quanto a predição muda quando o valor do atributo muda. O resultado é normalizado e a soma da importância de todos os atributos é cem. A segunda é do tipo 'LossFunctionChange' e mostra a diferença no valor da função de perda do modelo com todos os atributos presentes para um modelo sem um atributo presente. Quanto maior essa diferença, mais importante é um atributo para a predição do modelo completo.

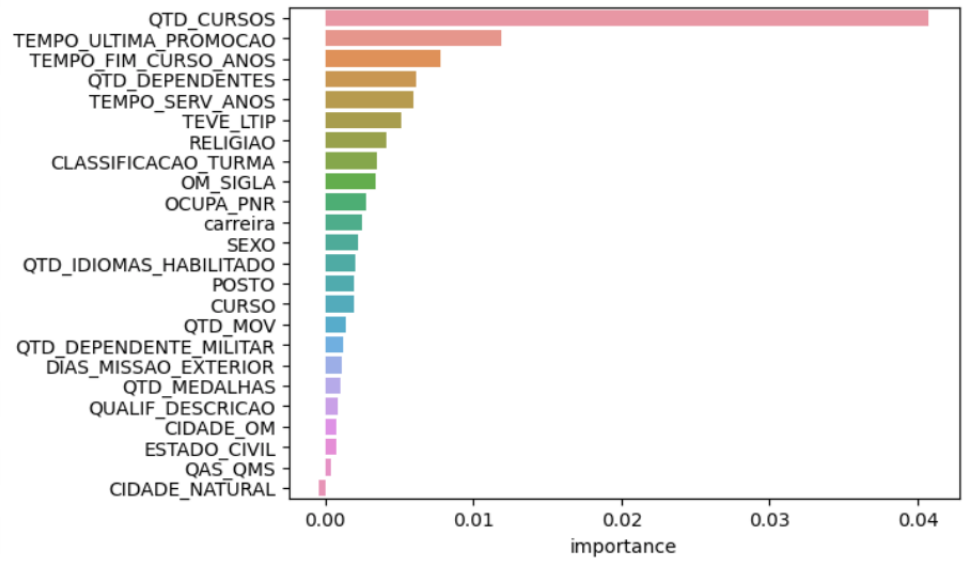
É interessante notar que o atributo mais relevante na classificação é a quantidade de cursos, independente da maneira que essa importância está sendo medida. A realização de cursos de aprimoramento no Exército não resulta apenas em aumento da satisfação pessoal com o trabalho, mas também tem o potencial de aumentar significantemente o salário de um oficial, o que poderia ser um dos motivadores para a alta importância. A quantidade de cursos realizados também está relacionada ao tempo que o oficial tem de carreira no Exército, assim como os três atributos seguintes com maior importância do tipo 'PredictionValueChange'. Isso ajuda a corroborar para a percepção saída da análise descritiva dos dados de que, quanto mais tempo a pessoa já está na carreira militar, menor é a chance dela pedir demissão.

Em seguida, no gráfico (a), aparecem os dois primeiros atributos categóricos da lista, religião e Organização Militar (OM) em que o militar está servindo. Religião ter uma importância tão alta é curioso, pois dos diversos atributos utilizados na classificação esse é um que não é considerado nos relatórios normalmente solicitados pelos gestores de recursos humanos do Exército. Quanto à OM do militar, por outro lado, já havia alguma expectativa de sua relevância para essa classificação.

A carreira do militar é uma característica que se esperava ter grande relevância para a decisão de sair, e o atributo relacionado a ela, 'QUALIF_DESCRICAO', é o nono na lista de importância dos atributos no gráfico (a), seguido pelo atributo 'QAS_QMS', que



(a) Importância com tipo 'PredictionValueChange'



(b) Importância com tipo 'LossFunctionChange'

Figura 5.4: Importâncias dos atributos na classificação realizada pelo CatBoost sem pré-processamento

apresenta uma subdivisão da qualificação para algumas das carreiras, como por exemplo a especialização de um oficial do QCO ou do QEM. No gráfico (b), por outro lado, esses dois atributos aparecem mais abaixo na importância, mas o atributo 'carreira' está aproximadamente no meio da lista, assim como o QAS_QMS no gráfico (a).

A figura 5.5 mostra um terceiro tipo de cálculo da importância dos atributos, os SHAP Values. Nesse cálculo, a contribuição de cada atributo e cada objeto é analisada. É possível visualizar, para os atributos numéricos, a distribuição dos valores altos ou

baixos pelos objetos e suas classificações. Se o SHAP value for positivo, a classificação é 'demitido'.

Assim, percebe-se que para a quantidade e cursos, uma menor quantidade está mais associada ao pedido de demissão. O mesmo vale para a quantidade de idiomas habilitados e para a quantidade de dependentes. O tempo de fim de curso já não tem um padrão claro a ser identificado, com valores baixos presentes nas duas extremidades da classificação. A classificação da turma traz uma informação interessante por mostrar que os militares com menor valor na classificação, ou seja, os melhores classificados da turma, são os com maior tendência a pedir demissão.

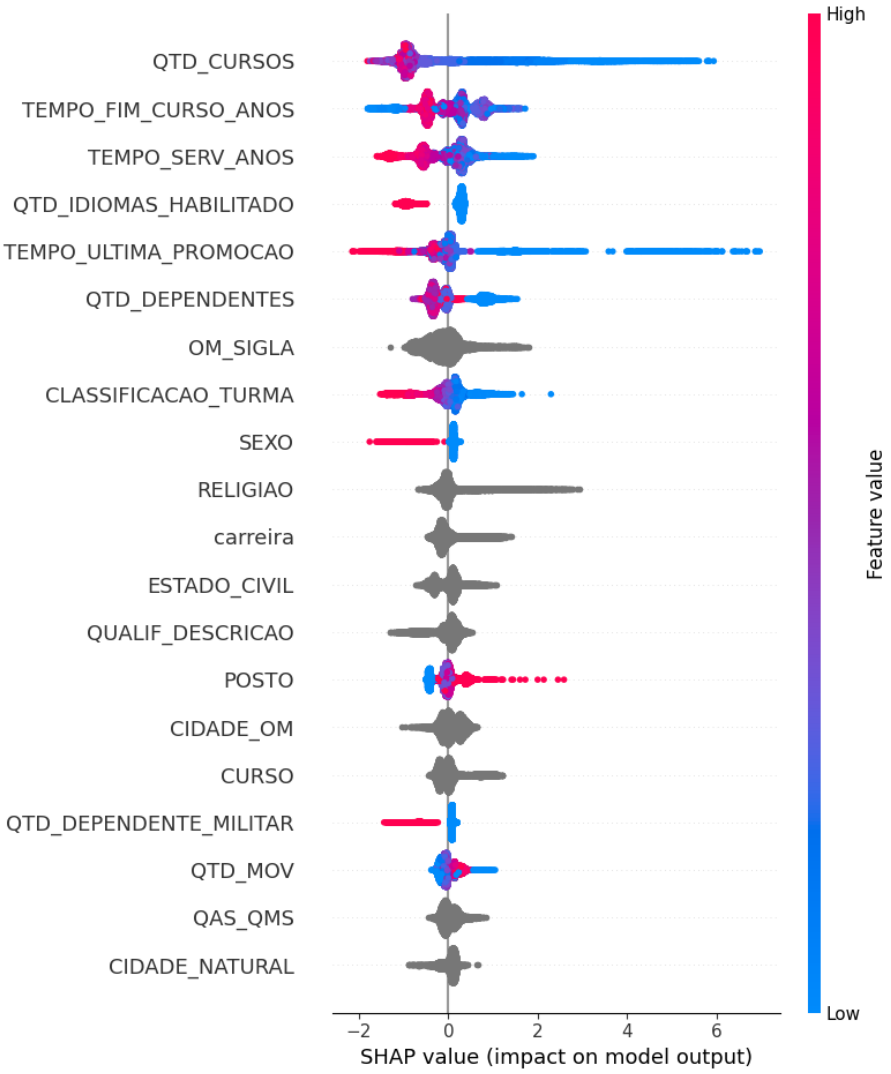


Figura 5.5: Importância dos atributos na classificação realizada pelo CatBoost sem pré-processamento com método SHAP Values

5.5.2 Conjunto de dados separado por carreira

Sabe-se que as diferentes carreiras têm taxas de evasão bem distintas entre elas, pois suas diversas características têm capacidade de influenciar as motivações para o pedido de demissão. Além disso, os concursos utilizados para admissão das carreiras são independentes e ter uma melhor predição da evasão para cada uma delas pode ser vantajoso para o planejamento do Exército Brasileiro. Dessa forma, decidiu-se analisar o resultado da predição feita sobre o conjunto de dados de validação com um dos melhores modelos encontrados na primeira etapa, que foi o CatBoost sem as etapas de pré-processamento, separando os dados por carreira.

A separação das carreiras foi feita, inicialmente, pelo valor do atributo 'QUALIF_DESCRICAO' do conjunto de dados, que representa a qualificação do oficial. Na tabela 5.15, as predições obtidas com o CatBoost no conjunto de dados completo são mostradas separadas por qualificação. A tabela mostra a quantidade de dados presente por qualificação no subconjunto de validação; a quantidade de casos de demitidos real; a quantidade de demitidos previsto; as quantidades de verdadeiros positivos (TP), falsos positivos (FP), falsos negativos (FN) e verdadeiros negativos (TN); os cálculos das métricas de *recall*, precisão e *f-score* para a classe "*sim*", dos demitidos; e os cálculos de acurácia dessa previsão e do caso dummy, ou seja, aquele que prevê apenas a classe "*não*" para todos os elementos.

QUALIFICACAO	TOTAL	DEMITIDOS REAL	DEMITIDOS PREDICAO	TP	FP	FN	TN	Recall ("sim")	Precision ("sim")	F-score ("sim")	acurácia	dummy
Artilharia	273	12	8	8	0	4	261	0,67	1,00	0,80	0,99	0,96
Cavalaria	274	11	6	5	1	6	262	0,45	0,83	0,59	0,97	0,96
Comunicações	143	4	1	1	0	3	139	0,25	1,00	0,40	0,98	0,97
Dentista	77	1	1	0	1	1	75	0,00	0,00		0,97	0,99
Engenharia	184	7	6	5	1	2	176	0,71	0,83	0,77	0,98	0,96
Farmaceutico	50	1	1	1	0	0	49	1,00	1,00	1,00	1,00	0,98
Infantaria	618	13	11	9	2	4	603	0,69	0,82	0,75	0,99	0,98
Intendência	278	22	18	17	1	5	255	0,77	0,94	0,85	0,98	0,92
Material Bélico	158	13	10	10	0	3	145	0,77	1,00	0,87	0,98	0,92
Médico	355	59	50	47	3	12	293	0,80	0,94	0,86	0,96	0,83
Quadro Complementar de Oficiais	420	31	24	24	0	7	389	0,77	1,00	0,87	0,98	0,93
Quadro de Engenheiros Militares	278	113	110	103	7	10	158	0,91	0,94	0,92	0,94	0,59

Tabela 5.15: Predições do conjunto de validação separadas por qualificação

Ao utilizar o teste de McNemar nesses resultados, comparando a predição feita com o resultado do algoritmo dummy, foi possível descobrir que as qualificações Artilharia, Intendência, Material Bélico, Médico, Quadro Complementar de Oficiais e Quadro de Engenheiros Militares tiveram resultados estatisticamente diferentes entre os dois algoritmos. Dessa forma, pode-se dizer que o modelo encontrado não é tão bom para as outras quali-

ficações. No entanto, a quantidade de dados presente no conjunto de dados de validação é pequena quando separado por qualificação e pode influenciar esse resultado.

Numa segunda tentativa de avaliar os resultados por carreira, as qualificações foram agrupadas de acordo com a escola de formação a que pertencem, resultando nas carreiras QCO, que engloba apenas a qualificação QCO; QEM, englobando apenas a qualificação QEM; Saúde, englobando as qualificações de dentista, médico e farmacêutico; e Combatente, englobando as demais qualificações. A tabela 5.16 mostra as mesmas informações da tabela 5.15, porém agora agrupadas dessa nova maneira.

CARREIRA	TOTAL	DEMITIDOS REAL	DEMITIDOS PREDICAO	TP	FP	FN	TN	Recall ("sim")	Precision ("sim")	F-score ("sim")	acurácia	dummy
Combatente	1928	82	60	56	4	26	1842	0,68	0,93	0,79	0,98	0,96
QCO	420	31	22	22	0	9	389	0,71	1,00	0,83	0,98	0,93
QEM	278	113	110	103	7	10	158	0,91	0,94	0,92	0,94	0,59
Saúde	482	61	51	48	3	13	418	0,79	0,94	0,86	0,97	0,87

Tabela 5.16: Predições do conjunto de validação separadas por carreiras

Para esse caso é possível ver na tabela que a quantidade de dados já é maior para cada carreira mesmo no conjunto de validação, que contém apenas 20% dos dados, permitindo uma melhor utilização do teste de McNemar. O resultado desse teste ao comparar as predições com o Dummy Classifier indica que as predições para todas as carreiras são estatisticamente diferentes de apenas classificar como a classe "*não*", ou seja, a predição feita no conjunto de dados completo já apresenta resultados úteis para cada carreira de forma separada.

Numa tentativa de tentar melhorar esse resultado por carreira, o primeiro e o segundo conjunto de testes que foram feitos nos dados completos também foram executados para o conjunto de dados separados por carreiras agrupadas, seguindo o mesmo processo feito para o conjunto de dados completo. O terceiro conjunto de testes, que utilizou algoritmos de detecção de *outliers*, não foi executado por ter apresentado resultados para o conjunto de dados completos que indicaram que os demissionários não podem ser tratados como *outliers*.

Tendo como base os valores de *recall* e acurácia apresentados na tabela 5.16, nenhuma combinação de algoritmos e pré-processadores de nenhum dos dois grupos de testes foi capaz de apresentar um resultado superior ao já obtido. A utilização do CatBoost sem as etapas de pré-processamento foi o algoritmo que mais se aproximou dos resultados obtidos anteriormente. Os valores das métricas encontradas nesse caso são mostrados na tabela 5.17.

Mesmo sendo o resultado mais próximo, é possível ver que o valor do *recall* para a classe "*sim*" é bem abaixo do obtido com o conjunto de dados completo, especialmente para a carreira do QCO. Com isso, pode-se supor que há suficientes similaridades entre

Carreira	classe	acurácia	precisão	recall	f-score
Combatente	não	0,98	0,99	1,00	0,99
	sim		0,94	0,68	0,79
QCO	não	0,96	0,96	1,00	0,98
	sim		0,94	0,47	0,63
QEM	não	0,93	0,91	0,98	0,95
	sim		0,97	0,86	0,91
Saúde	não	0,96	0,96	0,99	0,98
	sim		0,95	0,73	0,83

Tabela 5.17: Métricas obtidas com a execução do CatBoost sem pré-processamento nos conjuntos de dados separados por carreiras

os demissionários de diferentes carreiras que tornam a utilização do conjunto de dados completo uma melhor fonte para a modelagem desse problema quando comparado à utilização de conjuntos de dados menores, principalmente por causa da quantidade de dados nos conjuntos de dados separados.

Uma análise interessante resultante da execução do CatBoost com os conjuntos de dados separados que ajuda a corroborar essa ideia é a da importância dos atributos para cada carreira. A figura 5.6 mostra os gráficos obtidos para cada modelo gerado utilizando a importância do tipo 'PredictionValueChange'.

Pode-se perceber que a quantidade de cursos segue sendo o atributo de maior importância para todas as carreiras. Os tempos de fim de curso, tempo desde a última promoção, tempo de serviço e a religião seguem aparecendo com altas importâncias para todas as carreiras, ainda que as ordens entre eles sejam diferentes para cada uma delas. O atributo 'QUALIF_DESCRICAO', se torna o menos importante para as carreiras que só têm um valor possível nesse atributo, QCO e QEM, mas tem comportamentos bem diferentes para as outras duas carreiras. Esse atributo é pouco importante nas carreiras combatentes, só ganhando de 'SEXO' e 'OCUPA_PNR', enquanto segue tendo uma importância mediana para as carreiras de saúde. Nas carreiras QCO, QEM e Saúde, o atributo 'QAS_QMS', que distingue as especializações dentro da mesma carreira, aparece com uma importância maior do que no conjunto de dados completo.

5.6 Implantação

A última etapa do CRISP-DM é a implantação. Após a obtenção de um modelo capaz de realizar a predição desejada, ele deve ser utilizado de fato para poder trazer os benefícios propostos. Além disso, conforme novos dados são coletados, é possível que o modelo seja atualizado para se adaptar a qualquer mudança no padrão de detecção de evasão.

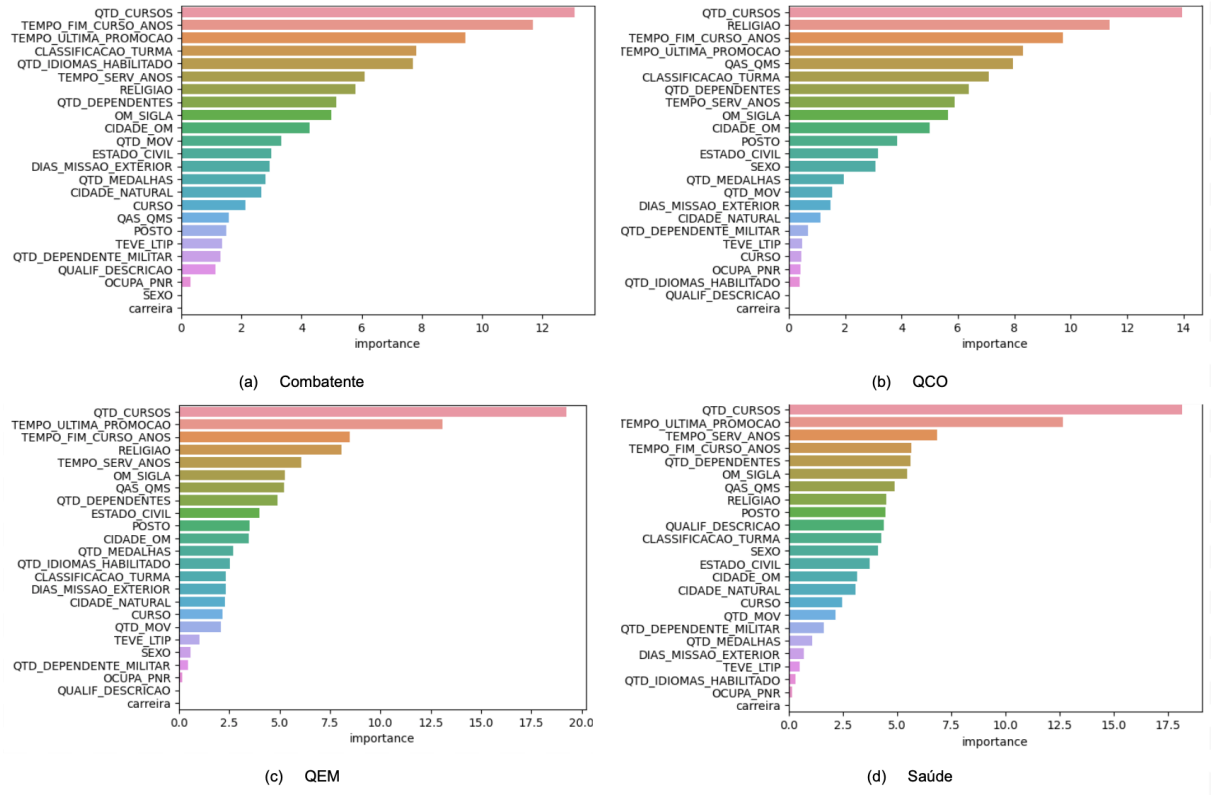


Figura 5.6: Importância dos atributos na classificação realizada pelo CatBoost sem pré-processamento por carreira

No Exército Brasileiro está sendo criada uma esteira de aprendizado de máquina, ou esteira MLOps, para facilitar a implantação de qualquer modelo gerado. Nessa esteira, é definido um processo de implantação de modelos e é disponibilizada a infraestrutura necessária para automatizar as etapas de extração e preparação dos dados e a execução de modelos, facilitando assim o trabalho dos cientistas de dados. O melhor modelo encontrado nesse trabalho, que foi o obtido com o algoritmo CatBoost sem pré-processamento para o conjunto de dados completo, será adaptado para ser utilizado nessa esteira, servindo como um dos casos testes da esteira definida.

A figura 5.7 mostra de forma resumida o que deve ser feito para treinar e implantar o modelo na esteira de aprendizado de máquina e como os novos dados devem ser tratados.

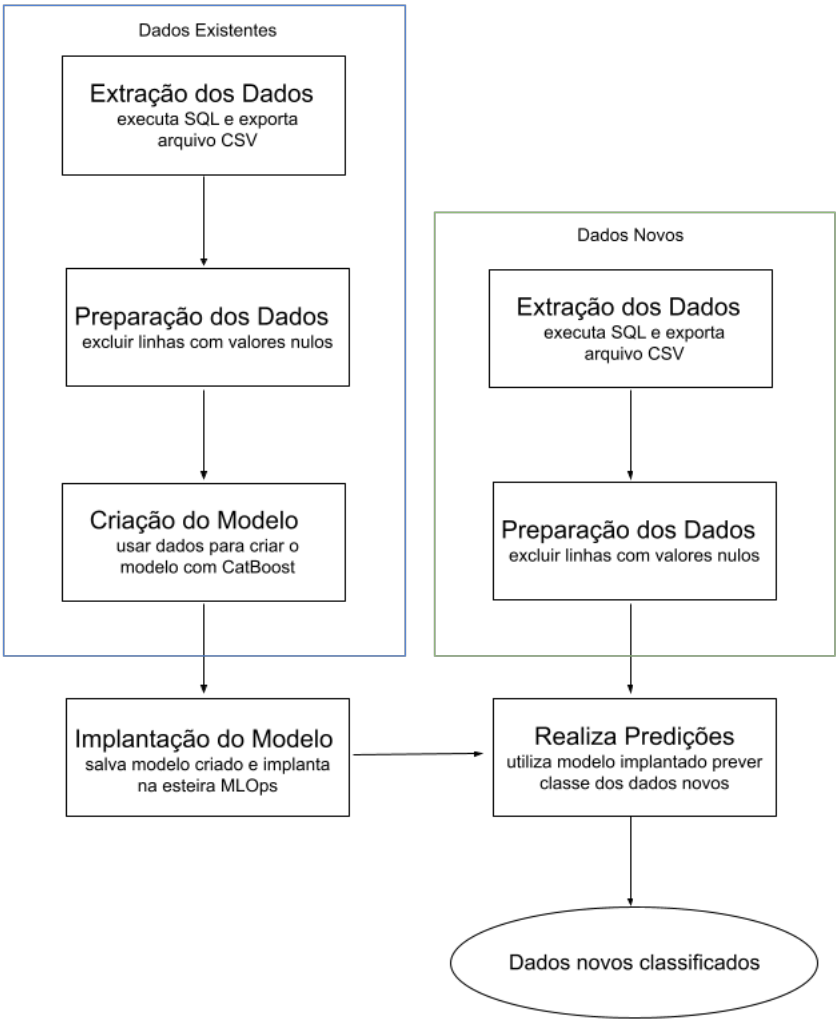


Figura 5.7: Esquema de treinamento e implantação do melhor modelo encontrado e do de como novos dados serão incorporados.

Capítulo 6

Conclusão

A perda de pessoal capacitado para o mercado civil que o Exército Brasileiro vem sofrendo tem afetado a capacidade de planejar e executar a sua missão constitucional, impactando a implantação de projetos como o CCOp Mv. Iniciativas de entender o fenômeno através de estatística descritiva já foram feitas, mas até o momento nenhuma iniciativa havia tentado realizar a predição de quais militares pediriam a demissão voluntária. Essa informação tem o potencial de facilitar a gestão de recursos humanos do EB, uma vez que permitiria o preparo antecipado de substitutos para oficiais que estejam ocupando funções de alta importância para essa Força Armada ou até mesmo poderia permitir que medidas sejam tomadas para evitar a demissão. Esse trabalho teve por objetivo encontrar um modelo preditivo que atendesse essa necessidade, identificando um padrão de evasão de oficiais do Exército Brasileiro.

Através da metodologia de revisão sistemática da literatura TEMAC, foi possível encontrar trabalhos relacionados que tentaram prever a evasão de empregados em empresas de diversas áreas de atuação. Descobriu-se que esse problema é de grande relevância para gerentes de recursos humanos há muito tempo, sendo possível encontrar pesquisas sobre o assunto sendo publicadas desde a década de 1920. Ainda assim, a abordagem computacional, mais especificamente utilizando técnicas de aprendizado de máquina, só passou a ser utilizada nas últimas duas décadas e teve uma quantidade de publicações crescente nos últimos cinco anos. A partir dos resultados encontrados nos trabalhos relacionados, entendeu-se quais as principais técnicas utilizadas para resolver esse problema e quais já foram utilizadas com sucesso em outros estudos de caso. Percebeu-se que não há uma abordagem universal que resolva o problema e cada caso deve ser analisado individualmente, principalmente devido à variação na quantidade e no tipo de dados disponíveis para cada estudo de caso. Assim, definiu-se um estudo de caso para encontrar qual a melhor abordagem para resolver o problema específico do Exército Brasileiro.

No estudo de caso realizado foi utilizado o *framework* CRISP-DM para aplicação de

projetos de aprendizado de máquina. Seguindo as etapas desse *framework*, passou-se pela etapa de entendimento dos negócios, na qual foi possível entender quais características e particularidades das diversas carreiras possíveis para os oficiais do Exército Brasileiro poderiam afetar a decisão de um militar pedir demissão para. A seguir, passou-se pela etapa de entendimento dos dados, que permitiu identificar atributos no banco de dados de pessoal do EB que fossem relacionados a essas características. A partir dos dados extraídos, detalhou-se o problema da evasão atualmente no Exército com estatística descritiva, percebendo que oficiais têm maior tendência a pedir demissão nos doze primeiros anos da carreira e que a maior porcentagem das evasões ocorrem com os militares do Quadro de Engenheiros Militares e com os médicos.

Os dados extraídos foram então tratados de diversas formas. Realizou-se o cálculo de atributos derivados dos armazenados originalmente, como o tempo passado desde a ocorrência de um evento e a contagem da ocorrência desses eventos. Lidou-se com dados faltantes em alguns atributos selecionados, realizando a remoção das linhas com dados faltantes ou a remoção do atributo do conjunto de dados, dependendo de quantas linhas eram afetadas pela presença de dados nulos. Avaliou-se formas de realizar a codificação de atributos categóricos e formas de normalizar os atributos numéricos, com a seleção de três técnicas a serem testadas para cada uma dessas transformações.

Na etapa de modelagem do CRISP-DM, foram realizados três conjuntos de testes. No primeiro, foram testados os algoritmos de classificação KNN, árvore de decisão, random forest, gradient boosting, extreme gradient boosting e CatBoost. Nos testes, todas as combinações entre algoritmos e técnicas de pré-processamento foram avaliadas, variando os hiperparâmetros de cada algoritmo até encontrar a combinação que apresentasse o melhor modelo. No segundo conjunto de testes, os mesmos algoritmos foram testados, porém dessa vez eles foram precedidos de técnicas de reamostragem, que aumentaram a presença da classe minoritária e reduziram a presença da classe majoritária, deixando o conjunto de dados mais balanceado. Por fim, no terceiro conjunto de testes, foram utilizados algoritmos de detecção de *outliers* para ver se a evasão poderia ser tratada como uma anomalia no conjunto de dados. Os mesmos conjuntos de testes foram executados para o conjunto de dados completo e também para o conjunto de dados separado por carreira. As carreiras testadas foram decididas baseadas na escola de formação de entrada dos oficiais, uma vez que a divisão por Quadro, Arma ou Serviço, que é a forma que o Exército lida com as carreiras, resultou em conjuntos de dados pequenos demais para a execução dos testes.

A etapa de avaliação do CRISP-DM comparou os resultados dos modelos através das métricas de acurácia, *recall*, precisão e *f-score*. Também foi utilizado o teste de McNemar para garantir que os resultados eram estatisticamente diferentes entre eles e que diferiam

de um preditor que sempre classificasse com a classe majoritária. A partir dessas análises, chegou-se à conclusão que os algoritmos XGBoost e CatBoost utilizados no conjunto de dados completo e sem a utilização de algoritmos de reamostragem, mas com etapas de pré-processamento, e também o algoritmo CatBoost sem etapas de pré-processamento obtiveram os melhores modelos para resolver esse problema no estudo de caso do Exército Brasileiro, que esse modelo é melhor do que um algoritmo que classifique sempre com a classe majoritária e que os modelos obtidos pelos algoritmos são estatisticamente iguais. Ainda foi possível analisar que os atributos que representam a quantidade de cursos realizados, o tempo desde o fim da formação e o tempo desde a última promoção são os três atributos com maior importância na classificação feita com o modelo gerado pelo CatBoost sem pré-processamento.

O modelo gerado está pronto para ser implantado na futura esteira de aprendizado de máquina que está sendo criada no Exército Brasileiro, permitindo que o modelo seja utilizado em novos dados e que os resultados da classificação sejam incorporados em painéis utilizados pelos gestores de recursos humanos do Exército para a tomada de decisão. Essa foi a primeira vez que uma iniciativa de utilização de análise preditiva para a resolução do problema de evasão de oficiais foi realizada e os resultados encontrados são capazes de prever com uma alta acurácia a alto *recall* a saída dos militares, principalmente ao se considerar o Quadro de Engenheiros Militares, que é a carreira mais afetada pela evasão. Além disso, os resultados mostraram informações sobre os atributos que mais influenciam no pedido de demissão que não haviam sido consideradas anteriormente pelos gestores de recursos humanos do Exército, como a importância alta da quantidade de cursos realizados e da religião. A implantação do modelo gerado é essencial para que os resultados atingidos levem a efetivos benefícios para o Exército Brasileiro.

Em trabalhos futuros, uma opção visualizada para melhorar os resultados obtidos é a aquisição de novos atributos através da realização de questionários. Dessa forma, outros aspectos da carreira militar que não são registrados nos bancos de dados de pessoal podem ser avaliados, como o nível de satisfação do oficial com a carreira.

Referências

[1] Chapman, Pete, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer e Rudiger Wirth: *Crisp-dm 1.0 step-by-step data mining guide*. Relatório Técnico, The CRISP-DM consortium, August 2000. x, 12

[2] *Scikit-learn - nearest neighbors classification*, 2022. `https://scikit-learn.org/stable/modules/neighbors.html#classification`. x, 27

[3] *Decision trees classification*, 2022. `https://scikit-learn.org/stable/modules/tree.html#classification`. x, 28

[4] Saradhi, V. Vijaya e Girish Keshav Palshikar: *Employee churn prediction*. Expert Systems with Applications, 38, março 2011, ISSN 09574174. 2, 22, 39

[5] Mariano, Ari e Maíra Santos: *Revisão da literatura: Apresentação de uma abordagem integradora*. AEDM International Conference–Economy, Business and Uncertainty: Ideas for a European and Mediterranean industrial policy, setembro 2017. 5, 8

[6] Muchinsky, P.M. e M.L. Tuttle: *Employee turnover: An empirical and methodological assessment*. Journal of Vocational Behavior, 14:43–77, 1979. 10, 21

[7] Lakatos, Eva Maria e Marina de Andrade Marconi: *Metodologia Científica*. Grupo GEN, 7ª edição, 2017. 11

[8] Wazlawick, Raul Sidnei: *Metodologia de Pesquisa para Ciência da Computação*. Campus-Elsevier, 2ª edição, 2014. 11

[9] Pereira, Adriana Soares, Dorlivete Moreira Shitsuka, Fábio José Pereira e Ricardo Shitsuka: *Metodologia da Pesquisa Científica*. UFSM, 2018. `https://repositorio.ufsm.br/bitstream/handle/1/15824/Lic_Computacao_Metodologia-Pesquisa-Cientifica.pdf`. 11

[10] Ahn, Jaehyun, Junsik Hwang, Doyoung Kim, Hyukgeun Choi e Shinjin Kang: *A survey on churn analysis in various business domains*. IEEE Access, 8:220816–220839, 2020, ISSN 21693536. 15, 18

[11] Madane, S. e D. Chitre: *A survey of employee and customer churn prediction methodologies*. Advances in Mathematics: Scientific Journal, 9:3955–3962, 2020, ISSN 18578438. 16

[12] Srivastava, P.R. e P. Eachempati: *Intelligent employee retention system for attrition rate analysis and churn prediction: An ensemble machine learning and multi- criteria decision-making approach.* Journal of Global Information Management, 29, 2021. 16

[13] Teng, M., H. Zhu, C. Liu e H. Xiong: *Exploiting network fusion for organizational turnover prediction.* ACM Transactions on Management Information Systems, 12, 2021. 16

[14] Teng, M., H. Zhu, C. Liu, C. Zhu e H. Xiong: *Exploiting the contagious effect for employee turnover prediction.* páginas 1166–1173, 2019, ISBN 9781577358091. 16

[15] Yuan, J.: *Research on employee turnover prediction based on machine learning algorithms.* páginas 114–120, 2021, ISBN 9780738131702. 16

[16] Joseph, R., S. Udupa, S. Jangale, K. Kotkar e P. Pawar: *Employee attrition using machine learning and depression analysis.* páginas 1000–1005, 2021, ISBN 9781665412728. 17

[17] Jain, N., A. Tomar e P.K. Jana: *A novel scheme for employee churn problem using multi-attribute decision making approach and machine learning.* Journal of Intelligent Information Systems, 56:279–302, 2021. 17

[18] Zhao, Yue, Maciej Hryniewicki, Francesca Cheng, Boyang Fu e Xiaoyu Zhu: *Employee Turnover Prediction with Machine Learning: A Reliable Approach*, páginas 737–758. janeiro 2019, ISBN 978-3-030-01056-0. 17, 21, 22, 27, 29

[19] Gao, Xiang, Junhao Wen e Cheng Zhang: *An improved random forest algorithm for predicting employee turnover.* MATHEMATICAL PROBLEMS IN ENGINEERING, 2019, 2019, ISSN 1024-123X. 17, 29

[20] Alduayj, S.S. e K. Rajpoot: *Predicting employee attrition using machine learning.* páginas 93–98, 2019, ISBN 9781538666739. 17, 27

[21] Ullah, I., H. Hussain, I. Ali e A. Liaquat: *Churn prediction in banking system using k-means, lof, and cblof.* 2019, ISBN 9781728138251. 18, 35

[22] Javed, Dr Sarfaraz e Ahmad Azhar: *Forecasting employee turnover for human resource based on time series analysis.* International Journal of Economic Research, 14:445–456, dezembro 2017. 18

[23] Zhu, Xiaojuan, William Seaver, Rapinder Sawhney, Shuguang Ji, Bruce Holt, Gurudatt Bhaskar Sanil e Girish Upreti: *Employee turnover forecasting for human resource management based on time series analysis.* JOURNAL OF APPLIED STATISTICS, 44:1421–1440, 2017, ISSN 0266-4763. 18

[24] Óskarsdóttir, María, Tine Van Calster, Bart Baesens, Wilfried Lemahieu e Jan Vanthienen: *Time series for early churn detection: Using similarity based classification for dynamic networks.* Expert Systems with Applications, 106:55–65, 2018, ISSN 0957-4174. `https://www.sciencedirect.com/science/article/pii/S0957417418302276`. 18

[25] Zais, M. e D. Zhang: *A markov chain model of military personnel dynamics*. International Journal of Production Research, 54:1863–1885, 2016. tenho acesso ao texto completo, mas só online. não me deixam baixar o pdf no site. 18

[26] Fallucchi, Francesca, Marco Coladangelo, Romeo Giuliano e Ernesto William De Luca: *Predicting employee attrition using machine learning techniques*. Computers, 9(4), 2020, ISSN 2073-431X. https://www.mdpi.com/2073-431X/9/4/86. 21, 22

[27] Micci-Barreca, Daniele: *A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems*. SIGKDD Explor. Newsl., 3(1):27–32, jul 2001, ISSN 1931-0145. https://doi.org/10.1145/507533.507538. 22, 23, 24

[28] Pargent, Florian, Florian Pfisterer, Janek Thomas e Bernd Bischl: *Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features*. Computational Statistics, 37(5):2671–2692, março 2022. https://doi.org/10.1007/s00180-022-01207-6. 23

[29] Hancock, John T. e Taghi M. Khoshgoftaar: *Survey on categorical data for neural networks*. Journal of Big Data, 7(1), abril 2020. https://doi.org/10.1186/s40537-020-00305-w. 24

[30] Singh, Dalwinder e Birmohan Singh: *Investigating the impact of data normalization on classification performance*. Applied Soft Computing, 97:105524, dezembro 2020. https://doi.org/10.1016/j.asoc.2019.105524. 25

[31] *Robust scaler*, 2022. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html#sklearn.preprocessing.RobustScaler. 26

[32] Goldschmidt, Ronaldo, Emmanuel Passos e Eduardo Bezerra: *Data Mining*. CGrupo GEN, 2ª edição, 2015. 27, 28

[33] Ruggieri, S.: *Efficient c4.5 [classification algorithm]*. IEEE Transactions on Knowledge and Data Engineering, 14(2):438–444, 2002. 28

[34] Dash, Shailey, novembro 2022. https://towardsdatascience.com/decision-trees-explained-entropy-information-gain-gini-index-ccp-pruning-4d78070db36c. 29

[35] Breiman, Leo. *Machine Learning*, 45(1):5–32, 2001. https://doi.org/10.1023/a:1010933404324. 29

[36] Bentéjac, Candice, Anna Csörgő e Gonzalo Martínez-Muñoz: *A comparative analysis of gradient boosting algorithms*. Artificial Intelligence Review, 54(3):1937–1967, agosto 2020. https://doi.org/10.1007/s10462-020-09896-5. 30, 31

[37] Friedman, Jerome H.: *Greedy function approximation: A gradient boosting machine.* The Annals of Statistics, 29(5), outubro 2001. https://doi.org/10.1214/aos/1013203451. 30

[38] Prokhorenkova, Liudmila, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush e Andrey Gulin: *Catboost: unbiased boosting with categorical features*, 2017. https://arxiv.org/abs/1706.09516. 31

[39] Chawla, N. V., K. W. Bowyer, L. O. Hall e W. P. Kegelmeyer: *Smote: Synthetic minority over-sampling technique*. 2011. https://arxiv.org/abs/1106.1813. 31, 32

[40] Han, Hui, Wen Yuan Wang e Bing Huan Mao: *Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning*. Em *Lecture Notes in Computer Science*, páginas 878–887. Springer Berlin Heidelberg, 2005. https://doi.org/10.1007/11538059_91. 32

[41] Nguyen, Hien M., Eric W. Cooper e Katsuari Kamei: *Borderline over-sampling for imbalanced data classification*. International Journal of Knowledge Engineering and Soft Data Paradigms, 3(1):4, 2011. https://doi.org/10.1504/ijkesdp.2011.039875. 33

[42] He, Haibo, Yang Bai, Edwardo A. Garcia e Shutao Li: *Adasyn: Adaptive synthetic sampling approach for imbalanced learning*. Em *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, páginas 1322–1328, 2008. 33

[43] Mani, Inderjeet e I Zhang: *knn approach to unbalanced data distributions: a case study involving information extraction*. Em *Proceedings of workshop on learning from imbalanced datasets*, volume 126, páginas 1–7. ICML, 2003. 33

[44] *Imbalanced-learn - under sampling - version 0.10.1*, 2022. https://imbalanced-learn.org/stable/under_sampling.html. 33, 34

[45] Sangeetha, T. e G.A. Mary: *A study on different methods of outlier detection algorithms in data mining*. International Journal of Computer Information Systems and Industrial Management Applications, 12:398–411, 2020. 34, 35

[46] Breunig, Markus M., Hans Peter Kriegel, Raymond T. Ng e Jörg Sander: *Lof: Identifying density-based local outliers*. SIGMOD Rec., 29(2):93–104, may 2000, ISSN 0163-5808. https://doi.org/10.1145/335191.335388. 35

[47] *Scikit-learn - outlier detection*, 2022. https://scikit-learn.org/stable/modules/outlier_detection.html. 35

[48] Liu, Fei Tony, Kai Ming Ting e Zhi Hua Zhou: *Isolation forest*. Em *2008 Eighth IEEE International Conference on Data Mining*, páginas 413–422, 2008. 36

[49] Bouckaert, Remco R. e Eibe Frank: *Evaluating the replicability of significance tests for comparing learning algorithms*. Em Dai, Honghua, Ramakrishnan Srikant e Chengqi Zhang (editores): *Advances in Knowledge Discovery and Data Mining*, páginas 3–12, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg, ISBN 978-3-540-24775-3. 36

[50] Dietterich, Thomas G.: *Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms*. Neural Computation, 10(7):1895–1923, outubro 1998, ISSN 0899-7667. https://doi.org/10.1162/089976698300017197. 36, 37

[51] Brasileiro, Exército: *Armas, quadros e serviços*. https://www.eb.mil.br/armas-quadros-e-servicos, acesso em 2021-12-29. 38

[52] McGinnis, Will: *Category encoders*, 2022. https://contrib.scikit-learn.org/category_encoders/index.html. 46