UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

LEONILDO COSTA E SILVA

APRENDIZADO DE MÁQUINA COM TREINAMENTO CONTINUADO APLICADO À PREVISÃO DE DEMANDA DE CURTO PRAZO: O CASO DO RESTAURANTE UNIVERSITÁRIO DA UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Uberlândia
2019

LEONILDO COSTA E SILVA

APRENDIZADO DE MÁQUINA COM TREINAMENTO CONTINUADO APLICADO À PREVISÃO DE DEMANDA DE CURTO PRAZO: O CASO DO RESTAURANTE UNIVERSITÁRIO DA UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal de Uberlândia como requisito parcial para obtenção do título de Mestre em Ciências.

Orientador: Prof. Dr. Aniel Silva de Morais

Coorientador: Prof. Dr. Josué Silva de Morais

Uberlândia

2019

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

S586 2019

Silva, Leonildo Costa e, 1978-
Aprendizado de máquina com treinamento continuado aplicado à previsão de demanda de curto prazo [recurso eletrônico] : o caso do Restaurante Universitário da Universidade Federal de Uberlândia / Leonildo Costa e Silva. - 2019.

Orientador: Aniel Silva de Morais.
Coorientador: Josué Silva de Morais.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Engenharia Elétrica.
Modo de acesso: Internet.
Disponível em: http://dx.doi.org/10.14393/ufu.di.2019.2001
Inclui bibliografia.
Inclui ilustrações.

1. Engenharia elétrica. I. Morais, Aniel Silva de, 1979-, (Orient.). II. Morais, Josué Silva de, 1981-, (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Engenharia Elétrica. IV. Título. CDU: 621.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

LEONILDO COSTA E SILVA

APRENDIZADO DE MÁQUINA COM TREINAMENTO CONTINUADO APLICADO À PREVISÃO DE DEMANDA DE CURTO PRAZO: O CASO DO RESTAURANTE UNIVERSITÁRIO DA UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal de Uberlândia como requisito parcial para obtenção do título de Mestre em Ciências.

Uberlândia, 27 de Maio de 2019.

BANCA EXAMINADORA

Prof. Dr. Aniel Silva de Morais – Orientador
Universidade Federal de Uberlândia – UFU

Prof. Dr. Josué Silva de Morais – Coorientador
Universidade Federal de Uberlândia – UFU

Prof. Dr. Kleber Lopes Fontoura
Centro Federal de Educação Tecnológica de Minas Gerais – CEFET-MG

Prof. Dr. Keiji Yamanaka
Universidade Federal de Uberlândia – UFU

Dedicado aos meus pais Leonildo e Suely, em reconhecimento aos esforços incondicionais devotados em minha formação pessoal e profissional.

AGRADECIMENTOS

Agradeço a Deus, por tudo.

Aos meus avós maternos, Maria e Lázaro, que de alguma forma sempre estiveram comigo nesta jornada.

À minha filha Alice, pelos desenhos inspiradores.

À minha querida esposa Lucélia, pela dedicação incondicional à nossa família. Pela cumplicidade. Por me fazer seguir em frente nos momentos de fraqueza.

Ao meu orientador Prof. Dr. Aniel Silva de Morais, pela confiança, amizade e parceria de sempre.

Ao meu co-orientador Prof. Dr. Josué Silva de Morais por me transmitir com sabedoria e serenidade conhecimentos importantíssimos na área de inteligência artificial. Pelo incentivo de sempre. Pela cobrança por resultados, quando necessário.

Ao corpo docente e técnico administrativo da Faculdade de Engenharia Elétrica da UFU.

Aos colegas de trabalho Guilherme Palhares Theodoro e Keynes Masayoshi Kanno pelo fornecimento de dados fundamentais para a execução deste trabalho.

Ao diretor do Centro de Tecnologia de Informação da UFU, Prof. Dr. Luís Fernando Faina, e ao coordenador da Divisão de Sistemas, Dr. Raulcézar Maximiano Figueira Alves, pela concessão do meu afastamento das atividades laborais, tornando possível a minha participação neste curso de pós-graduação.

A todos que contribuíram para a realização deste sonho, meus sinceros agradecimentos.

RESUMO

Planejar em situações de incerteza requer o uso de informações que auxiliam os gestores na tomada de decisão. A análise de séries temporais com o objetivo de criar previsões sobre eventos futuros é uma importante aliada neste processo. Neste contexto situa-se a análise de previsão de demanda que pode ser definida como uma projeção do futuro sobre a procura por bens ou serviços oferecidos por uma organização. De posse dessa informação os gestores podem orientar suas ações em função dos objetivos operacionais e estratégicos. Portanto, gerar previsões confiáveis e precisas é uma atividade indispensável para qualquer instituição com foco no fornecimento de bens ou serviços. Este trabalho investiga a aplicação de métodos baseados em aprendizado de máquina para a geração de previsões quantitativas de curto prazo. Dada a volatilidade característica das relações de consumo influenciadas por diversos fatores, a relação entre as variáveis explicativas e a demanda prevista podem sofrer alterações ao longo do tempo. Assim, os modelos de previsão podem apresentar degradação da sua acurácia se não forem atualizados constantemente. Dessa forma, esta pesquisa compara o desempenho de quatro algoritmos de aprendizado de máquina com capacidade de treinamento continuado. O *dataset* utilizado no experimento consiste na demanda diária de refeições do restaurante universitário da Universidade Federal de Uberlândia. Para treinamento dos modelos foram coletados dados compreendidos entre janeiro de 2015 a abril de 2018, sendo que os últimos seis meses foram reservados para o procedimento de teste. A metodologia adotada baseia-se no processo CRISP-DM com utilização de práticas apropriadas a projetos de aprendizado de máquina. Os resultados indicaram que a metodologia empregada foi capaz de produzir modelos com boa qualidade de ajuste (resíduos não correlacionados e média residual próxima de zero) além de uma capacidade de generalização adequada, aferida pelas métricas RMSE e MAE.

Palavras-chave: planejamento. aprendizado de máquina. treinamento continuado. teoria da ressonância adaptativa. previsão de demanda. redes neurais artificiais.

ABSTRACT

Planning in situations of uncertainty requires the use of information that assists managers in decision making. The analysis of time series with the aim of creating predictions about future events is an important ally in this process. In this context the demand forecast analysis can be defined as a projection of the future on the demand for goods or services offered by an organization. With this information, managers can guide their actions in function of the operational and strategic objectives. Therefore, generating reliable and accurate forecasts is an indispensable activity for any institution focused on the supply of goods or services. This work investigates the application of methods based on machine learning for the generation of short term quantitative forecasts. Given the volatility characteristic of consumer relations influenced by several factors, the relationship between the explanatory variables and the forecast demand may change over time. Thus, forecasting models may show degradation of their accuracy if they are not constantly updated. Thus, this research compares the performance of four machine learning algorithms with continuous training capacity. The dataset used in the experiment consists of the daily demand for meals at the university restaurant of the Federal University of Uberlândia. To train the models, data were collected between January 2015 and April 2018, and the last six months were reserved for the test procedure. The methodology adopted is based on the CRISP-DM process with the use of appropriate practices for machine learning projects. The results indicated that the methodology employed was able to produce models with good quality of fit (uncorrelated residues and residual mean close to zero) and an adequate generalization capacity, measured by RMSE and MAE metrics.

Keywords: planning. machine learninig. continuous learning. adaptive resonance theory. demand forecasting. artificial neural network.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ART *Adaptive Resonance Theory*

CRISP-DM *Cross Industry Standard Process for Data Mining*

CSV *Comma-separated values*

CTI Centro de Tecnologia da Informação

EA *Euclidean ART*

EAM *Euclidean ARTMAP*

ERAVC Eliminação recursiva de atributos com validação cruzada

FA *Fuzzy ART*

FAM *Fuzzy ARTMAP*

FDP Função densidade de probabilidade

`GLORIA_ALMOCO` Série temporal histórica da quantidade diária de refeições servidas no almoço do RU Glória

HTML *Hyper Text Markup Language*

JSON *Java Script Object Notation*

KNN *K-Nearest Neighbors*

MAE *Mean Absolute Error*

MEV Modelo de Espaço Vetorial

PCA *Principal Component Analysis*

`PONTAL_ALMOCO` Série temporal histórica da quantidade diária de refeições servidas no almoço do RU Pontal

`PONTAL_CAFE` Série temporal histórica da quantidade diária de refeições servidas no café da manhã do RU Pontal

`PONTAL_JANTAR` Série temporal histórica da quantidade diária de refeições servidas no jantar do RU Pontal

Quant. Quantidade

RMSE *Root Mean Square Error*

RNA Rede Neural Artificial

RU Restaurante universitário

SG Sistema de Gestão

SNAIVE *Seasonal Naive*

SQL *Structured Query Language*

STAMONICA_ALMOCO Série temporal histórica da quantidade diária de refeições servidas no almoço do RU Santa Mônica

STAMONICA_CAFE Série temporal histórica da quantidade diária de refeições servidas no café da manhã do RU Santa Mônica

STAMONICA_JANTAR Série temporal histórica da quantidade diária de refeições servidas no jantar do RU Santa Mônica

TF-IDF *Term Frequency-Inverse Document Frequency*

UFU Universidade Federal de Uberlândia

UMU_ALMOCO Série temporal histórica da quantidade diária de refeições servidas no almoço do RU Umuarama

UMU_CAFE Série temporal histórica da quantidade diária de refeições servidas no café da manhã do RU Umuarama

UMU_JANTAR Série temporal histórica da quantidade diária de refeições servidas no jantar do RU Umuarama

VC Validação Cruzada

XML *eXtensible Markup Language*

a_c**FERIADO** atributo feriado

a_c**FERIAS** atributo férias

a_c**GREVE** atributo greve

a_c**PRIMEIRA_SEMANA** atributo da primeira semana do semestre

a_c**SEGUNDA_VESPERA_FERIADO** atributo da segunda-feira véspera de feriado

a_c**SEXTA_POS_FERIADO** atributo da sexta-feira subsequente a um feriado

a_c**VESPERA_FERIADO_PROLONGADO** atributo dos dois dias que antecedem um feriado prolongado

a_c**VESPERA_FERIAS** atributo das duas semanas que antecedem as férias

a_{card} atributo cardápio

a_d**DIA_SEMANA** atributo dia da semana

a_d**MES** atributo mês

a_d**PUSA** atributo da primeira e última semana do ano

tsfresh *Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests*

SUMÁRIO

1 INTRODUÇÃO

Elaborar previsões é uma atividade que realizamos frequentemente em nossas vidas: estimamos o prazo para a conclusão de uma tarefa, a quantidade de material necessário para execução de uma obra ou analisamos o melhor investimento para as nossas economias. Ao realizar uma previsão estamos de alguma forma tentando reduzir incertezas sobre o futuro para que possamos tomar as melhores decisões no presente com a finalidade de alcançar algum objetivo.

No âmbito das organizações, seja ela fornecedora de bens ou serviços, a previsão assume um papel estratégico. Ter uma boa estimativa da demanda dos clientes por bens ou serviços, é essencial para o planejamento da produção. Com isso as organizações podem, por exemplo, determinar a quantidade de matéria-prima a ser adquirida, planejar a quantidade de bens prontos em estoque, dimensionar o número de pessoas a serem contratadas e até a quantidade de materiais de escritório a serem adquiridos. Como resultado, espera-se melhorar a eficiência da cadeia de suprimentos através da redução de custos, além do aprimoramento dos serviços aos clientes (CHASE, 2009).

Diante do exposto, fica evidente a importância de se produzir boas previsões. Neste sentido, segundo Kolassa e Siemsen (2016), o que determina a qualidade das previsões é o processo empregado para se obtê-las, e não o resultado dela isoladamente. O que deve ser avaliado é se o processo que gerou as previsões fez uso efetivo de todas as informações relevantes disponíveis na organização. Se a previsão ficou longe da demanda realizada, mas o processo que a gerou fez uso efetivo das informações, a organização não teve sorte. Por outro lado, se o processo desprezou informações importantes, mas mesmo assim a previsão ficou bem próxima da realidade, pode-se atribuir este fato a uma obra do acaso. Tal suposição reforça a ideia de que a previsão (baseada em modelos matemáticos) é uma projeção do futuro, baseada em dados históricos. Portanto, na impossibilidade de se prever o futuro diante da aleatoriedade dos fatos, quanto mais se conhecer sobre a variável que se deseja prever e fazer o uso efetivo desse conhecimento, maior será a probabilidade de se gerar boas previsões, que invariavelmente será acompanhada de um erro.

Além da importância do processo empregado na construção das previsões, a previsibilidade da série temporal também deve ser considerada para a avaliação das previsões. Se um modelo responde inadequadamente de forma recorrente, o motivo pode ser atri-

buído ao processo que o construiu, mas também pode ser devido ao excesso de ruído na série temporal. Assim, um bom processo de previsão não é aquele que torna a série temporal perfeitamente previsível, mas sim aquele que melhora a previsibilidade da série se comparado aos métodos mais simples (KOLASSA; SIEMSEN, 2016).

Uma vez que a demanda pode ser mensurada regularmente, o conjunto histórico dos valores obtidos podem ser considerados como uma série temporal. Como tal, a mesma é passível de análise para a elaboração de modelos matemáticos capazes de preverem um ponto no futuro baseado em padrões captados nos dados históricos. Outra abordagem utilizada para a previsão de demanda é através do julgamento de uma pessoa ou um grupo de pessoas. Neste caso, o gestor baseado na sua intuição e experiência, emite sua opinião sobre a demanda prevista. Este último método é denominado de método qualitativo enquanto o primeiro, quantitativo (CHASE, 2009).

Os métodos de previsão quantitativos podem ser divididos em dois grandes grupos: modelos de série temporais e modelos causais ou explicativos. Os métodos de séries temporais, consideram apenas as observações históricas da variável que se deseja prever. O objetivo é encontrar um modelo que melhor representa a série e então utiliza-lo para prever um ponto no futuro. Já os modelos explicativos consideram a relação entre a variável dependente (que se deseja prever) com uma ou mais variáveis independentes ou explicativas (HYNDMAN; ATHANASOPOULOS, 2018).

Dentre as técnicas clássicas de previsão, destacam-se os métodos estatísticos listados no Quadro 1. Cada método sugere um modelo baseado em um conjunto de suposições sobre como os dados são gerados. Geralmente estes modelos são compostos por um ou mais parâmetros que são estimados utilizando os dados históricos disponíveis (HYNDMAN; ATHANASOPOULOS, 2018).

Nos últimos anos observa-se um aumento significativo no uso de técnicas de aprendizado de máquina como alternativa aos métodos estatísticos de previsão de séries temporais. Este interesse é evidenciado pelo crescente número de artigos publicados em diversos veículos de publicações científicas de diversas áreas do conhecimento (Figura 1).

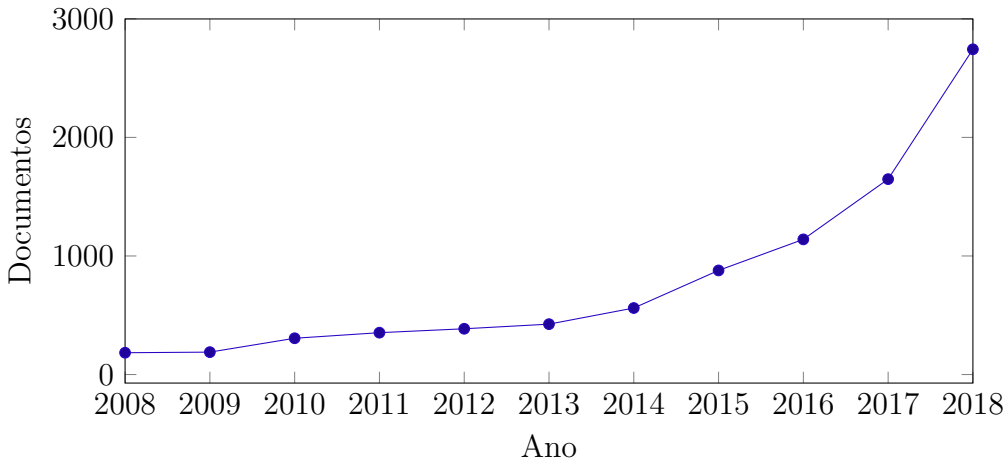
Dentro do universo de algoritmos de aprendizado de máquina, as Redes Neurais Artificiais (RNAs) apresentam características notáveis que as tornam interessantes para a tarefa de previsão. Segundo o teorema da aproximação universal (CYBENKO, 1989), as redes neurais podem aproximar qualquer função contínua de n variáveis reais. Esta

Quadro 1 – Métodos estatísticos clássicos de previsão

Método	**Descrição**
Modelos de série temporal	
Naive	Utiliza o último valor observado para prever o próximo ponto.
Média simples	Utiliza a média dos valores passados como previsão.
Média móvel simples	A previsão é baseada na média de n observações mais recentes.
Média móvel ponderada	A previsão é baseada na média de n observações mais recentes ponderadas por pesos. Os pesos decaem na medida que a observação fica mais distante.
Suavização exponencial	A previsão é baseada na combinação linear de todas observações históricas, ponderadas por pesos. Os pesos decaem exponencialmente na medida que a observação fica mais distante.
Suavização exponencial de Holt	Método de suavização exponencial para séries com tendência.
Suavização exponencial de Holt-Winter	Método de suavização exponencial para séries com tendência e sazonalidade.
Método Box-Jenkins	Metodologia para elaboração de modelos ARIMA (autoregressivos integrados e de médias móveis).
Modelos causais ou explicativos	
Regressão linear	Modela a relação linear entre duas variáveis utilizando o método dos mínimos quadrados.
Regressão linear múltipla	Semelhante à regressão linear, porém pode considerar mais de uma variável explicativa.

Fonte – Adaptado de Reid e Sanders (2015)

Figura 1 – Quantidade de publicações anuais sobre o tema de previsão com aprendizado de máquina. *String* de busca: (*forecasting* AND "*machine learning*").



Fonte – (SCOPUS, 2019)

capacidade é útil para capturar a complexidade de relacionamento entre variáveis de muitos problemas reais. Além disso, as RNAs são menos restritivas quanto às suposições sobre o processo de geração dos dados. Dessa forma são menos suscetíveis a erros de especificação de modelos se comparadas aos métodos estatísticos paramétricos. Em situações onde os dados são facilmente coletados, mas o mecanismo de geração dos mesmos é desconhecido, a aplicação das redes neurais torna-se vantajosa devido a sua capacidade de aprender com os dados (ZHANG, 2004).

Tradicionalmente, os modelos de previsão, sejam eles estatísticos ou de aprendizado de máquina, são ajustados de forma *offline* a partir de um conjunto de dados. Nesta abordagem assume-se que os dados disponíveis para treinamento representam completamente o processo que se deseja modelar. No entanto, esta suposição não é verdadeira para muitos problemas do mundo real. Em ambientes não estacionários, a relação entre as variáveis de entrada e a variável alvo podem mudar com tempo, resultando em uma degradação da acurácia do modelo. Esta mudança na função implícita que governa a geração dos dados é denominada de desvio de conceito (*concept drift*) e está associado a cenários onde ocorre um fluxo contínuo de dados (HOENS; POLIKAR; CHAWLA, 2012).

Uma das formas de superar os efeitos colaterais do desvio de conceito é através da atualização contínua do modelo assim que novos dados são disponibilizados. Tal abordagem é denominada de treinamento continuado ou aprendizado incremental. No entanto, para atingir este objetivo, o modelo deve ser capaz de lidar com uma importante questão: como assimilar rapidamente novas informações e ao mesmo tempo manter os conhecimentos adquiridos anteriormente? Grossberg (2013) denominou este problema como dilema estabilidade-plasticidade. Por outro lado, os modelos que não lidam com o dilema estabilidade-plasticidade, são acometidos pelo que Grossberg (2013) chama de esquecimento catastrófico. Situação onde os conhecimentos anteriores são perdidos frente a uma tentativa de aprender rapidamente uma nova informação.

1.1 O restaurante universitário

O Restaurante universitário (RU) da Universidade Federal de Uberlândia (UFU) serve mensalmente, em média, mais de 100 000 refeições entre café da manhã, almoço e jantar em quatro campi da universidade. Toda essa demanda é suprida através de um contrato celebrado com a empresa terceirizada Nutrir Refeições Coletivas Ltda. O contrato

prevê a "produção, transporte, disponibilização e distribuição de refeições e cafés da manhã, incluindo os serviços auxiliares de limpeza e higienização de áreas físicas, utensílios e equipamentos" a um custo anual de R$ 13.369.487,52 (UFU, 2018a).

Segundo a interpretação de Cardozo (1999) sobre o princípio da eficiência na administração pública (BRASIL, 1988):

> [...] pode-se definir esse princípio como sendo aquele que determina aos órgãos e pessoas da Administração Direta e Indireta que, na busca das finalidades estabelecidas pela ordem jurídica, tenham uma ação instrumental adequada, constituída pelo aproveitamento maximizado e racional dos recursos humanos, materiais, técnicos e financeiros disponíveis, de modo que possa alcançar o melhor resultado quantitativo e qualitativo possível, em face das necessidades públicas existentes (CARDOZO, 1999, p.166).

Em concordância com o princípio da eficiência, a administração do RU planeja mensalmente a operacionalização do fornecimento de refeições à comunidade universitária com o objetivo de racionalizar a utilização dos recursos públicos. Uma informação importante que baliza este planejamento é a previsão da demanda diária de refeições em um horizonte de 30 dias. Quando da elaboração deste trabalho, esta previsão era realizada qualitativamente, baseado no julgamento subjetivo da nutricionista do restaurante.

1.2 Objetivo

O objetivo geral deste trabalho consiste na elaboração de modelos baseados em aprendizado de máquina com treinamento continuado, com a tarefa de produzirem previsões de demanda do consumo diário de refeições do RU em um horizonte de 30 dias. Serão comparados os desempenhos dos seguintes algoritmos: *Fuzzy ARTMAP* (FAM), *Euclidean ARTMAP* (EAM), *K-Nearest Neighbors* (KNN) e *Seasonal Naive* (SNAIVE), sendo que os dois últimos foram utilizados como linha de base.

1.2.1 Objetivos específicos

- Análise exploratória da série histórica da demanda diária de refeições do RU, compreendida no período de Janeiro de 2015 à Abril de 2018.
- Preparação dos dados para ajuste dos modelos: limpeza, engenharia, seleção e transformação de atributos.

- Implementação e validação dos algoritmos FAM e EAM na linguagem Python.
- Otimização dos hiperparâmetros dos modelos.
- Avaliação dos modelos em regime de treinamento continuado: qualidade do ajuste e acurácia das previsões.
- Comparação dos resultados.

1.3 Organização

Este documento está organizado nos seguintes capítulos:

Capítulo 1 - Introdução: Este capítulo se inicia pela contextualização do tema abordado além de introduzir alguns conceitos iniciais. Posteriormente apresenta e justifica o problema que se pretente resolver, bem como os objetivos da pesquisa.

Capítulo 2 - Aprendizado de Máquina: Revisão da literatura e fundamentação teórica sobre o fluxo de atividades de projetos com aprendizado de máquina com ênfase no treinamento supervisionado e tarefas de regressão. Apresenta conceitos utilizados na pesquisa sobre a análise exploratória dos dados, engenharia, seleção e a transformação de atributos. Sob a perspectiva do dilema viés-variância, traz também as práticas mais difundidas para a otimização de hiperparâmetros com o objetivo de se obter modelos bem ajustados e com a melhor capacidade de generalização.

Capítulo 3 - Redes neurais artificiais ART: Fundamentação teórica utilizada no trabalho sobre as RNAs da família *Adaptive Resonance Theory* (ART). Apresenta o dilema da plasticidade-estabilidade, sendo esta uma característica indispensável aos modelos com capacidade de treinamento continuado. Descreve a arquitetura básica de uma rede ART e os seus desdobramentos em redes de treinamento não supervisionado e supervisionado. Detalha a arquitetura e algoritmos das redes FAM e EAM, ambas de treinamento supervisionado.

Capítulo 4 - Séries temporais: Apresenta a definição adotada nesse trabalho sobre séries temporais, seus componentes e propriedades.

Capítulo 5 - Material e Métodos: Detalhamento do fluxo de atividades aplicado ao problema de modelagem da série temporal da demanda diária de refeições do RU da UFU. Aborda desde o entendimento do negócio, coleta, análise exploratória e preparação dos dados, bem como o ajuste e avaliação dos modelos em regime de treinamento continuado. O capítulo ainda descreve o procedimento utilizado para a validação da implementação das redes FAM e EAM. O capítulo se encerra com a apresentação e discussão dos resultados obtidos na pesquisa.

Capítulo 6 - Considerações finais: Exposição das conclusões do trabalho evidenciando os principais resultados obtidos, limitações da pesquisa e trabalhos futuros.

2 APRENDIZADO DE MÁQUINA

Aprendizado de máquina ou *machine learning* é um ramo da Inteligência Artificial, que emprega técnicas e algoritmos na criação de modelos computacionais cuja característica principal é a capacidade de descobrir padrões num grande volume de dados ou de melhorarem o seu desempenho numa determinada tarefa através da experiência (SWAMYNATHAN, 2017; MOHRI; ROSTAMIZADEH; TALWALKAR, 2012). O pesquisador Arthur Lee Samuel (1959), considerado pioneiro na área de inteligência artificial (WIEDERHOLD; MCCARTHY, 1992), define aprendizado de máquina como o "campo de estudo que dá aos computadores a capacidade de aprender sem serem explicitamente programados"[1] (SIMON, 2013, p. 89, tradução nossa).

Uma definição mais precisa para o termo "aprender", no contexto de aprendizado de máquina foi dada por Mitchell (1997, p. 2, tradução nossa):

> Diz-se que um programa de computador aprende com a experiência E com relação a alguma classe de tarefas T e medida de desempenho P, se seu desempenho em tarefas em T, como medido por P, melhora com a experiência E [2].

Aprendizado de máquina tem sido aplicado na automação de atividades que para os humanos são executadas intuitivamente, mas que, no entanto, são difíceis de definir formalmente (SARKAR; BALI; SHARMA, 2017). Algumas aplicações típicas incluem (MOHRI; ROSTAMIZADEH; TALWALKAR, 2012):

a) classificação de texto ou documento, p.ex., deteção de *spam*;

b) processamento de linguagem natural;

c) reconhecimento e síntese de fala;

d) reconhecimento ótico de caracteres;

e) tarefas de visão computacional, p.ex., reconhecimento de imagem, deteção de faces;

f) deteção de fraudes (transações com cartão de crédito, telefone) e invasão de redes;

---

[1] *Field of study that gives computers the ability to learn without being explicitly programmed* (SIMON, 2013, p. 89)

[2] *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E* (MITCHELL, 1997, p. 2)

g) *games*;

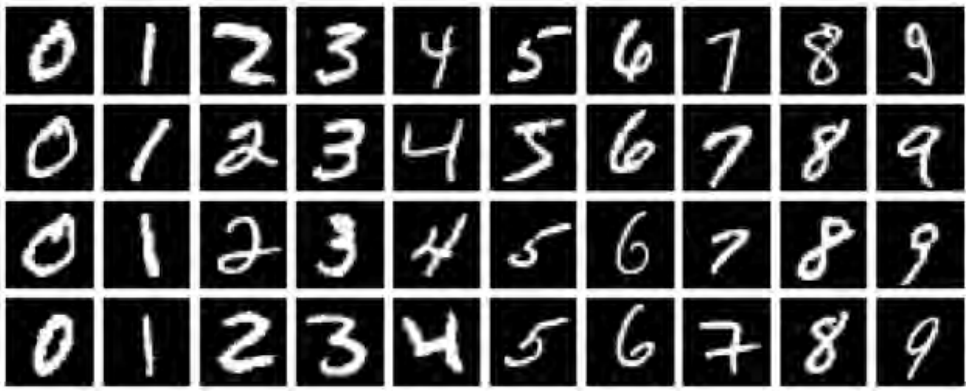
h) controle não assistido de veículos;

i) diagnósticos médicos;

j) sistemas de recomendações, motores de busca, sistemas de extração de informações.

Para exemplificar a utilização do aprendizado de máquina e introduzir alguns conceitos, considere o problema que consiste na elaboração de um programa de computador capaz de reconhecer dígitos escritos à mão como estes ilustrados na Figura 2.

Figura 2 – Amostras de dígitos escritos à mão



Fonte – Base de dados MNIST (LECUN; CORTES, 2010)

Cada dígito está representado eletronicamente por uma imagem com dimensões de 28×28 pixeis, onde cada um deles pode assumir valores inteiros do intervalo $[0, 255]$ (tons de cinza variando do preto ao branco). O procedimento deverá receber como entrada um vetor x de 784 elementos (pixeis da imagem) e então apresentar na saída o dígito correspondente $(0, 1, \ldots 9)$.

Utilizando uma abordagem tradicional, o procedimento poderia ser construído implementando-se regras baseadas nas formas dos traços dos dígitos. Devido ao grande número de possibilidades, inúmeras regras deverão ser escritas e inevitavelmente surgirá um dígito que não será enquadrado em nenhuma das regras elaboradas.

Na abordagem utilizando aprendizado de máquina, um modelo é construído através de um processo denominado treinamento ou ajuste. Neste procedimento é utilizado uma grande quantidade N de dígitos $\{x_1, x_2, \ldots, x_N\}$ chamado de conjunto de treinamento. Associado a cada padrão x_i, é conhecido antecipadamente o dígito que o mesmo repre-

senta. Este mapeamento é feito manualmente, analisando cada dígito e atribuindo a sua categoria pertencente $(0, 1, \ldots 9)$. No contexto de aprendizado de máquina esta informação é denominada de saída desejada, rótulo ou *target*, representado por y_i (BISHOP, 2006).

Durante o treinamento cada par (x_i, y_i) é apresentado gradativamente ao algoritmo de aprendizado de máquina, onde parâmetros internos (ou pesos) são ajustados com o objetivo de aprender a associação entre o vetor de entrada x_i e a sua saída correspondente y_i (SARKAR; BALI; SHARMA, 2017).

Após a fase de aprendizagem, o modelo será empregado para reconhecer novos dígitos não utilizados no procedimento de ajuste. Estes dados são denominados de conjunto de teste e são utilizados para medir a capacidade que o modelo possui de reconhecer corretamente padrões não utilizados no treinamento. Esta característica é conhecida como capacidade de generalização (BISHOP, 2006).

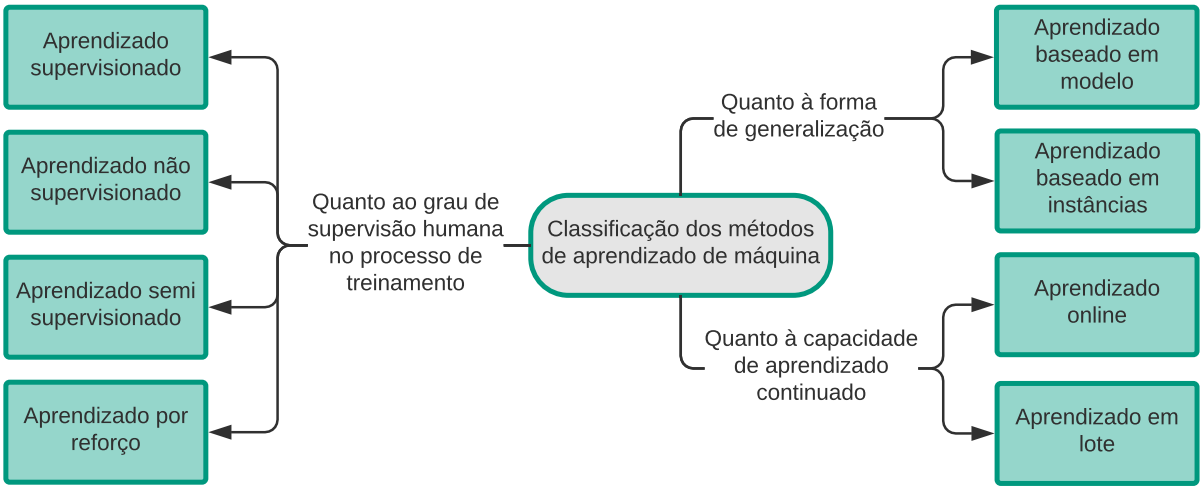
Aplicando a definição sugerida por Mitchell (1997) neste exemplo, a tarefa T é o reconhecimento e classificação de dígitos em imagens, a medida de desempenho P poderia ser o percentual de dígitos classificados corretamente e a experiência E é representada pelo banco de dados de dígitos escritos à mão (conjunto de treinamento).

2.1 Classificação dos métodos de aprendizado de máquina

Diante da diversidade de algoritmos e técnicas de aprendizado de máquina, classificá-los em categorias facilita a identificação dos métodos mais apropriados para cada tipo de aplicação. Nesta seção serão apresentadas as principais classificações dadas aos métodos de aprendizado de máquina (Figura 3) considerando três aspectos: grau de supervisão humana, forma de generalização e capacidade de aprendizado continuado. As categorias não são mutuamente exclusivas. Portanto, um método de aprendizado de máquina pode assumir uma classificação pertencente a cada um dos três aspectos considerados. A seguir é dada uma descrição das principais características de cada categoria:

a) **aprendizado supervisionado**: neste tipo de aplicação os vetores de entrada x_i são rotulados, ou seja, se conhece previamente o valor correspondente y_i. O objetivo do treinamento é criar uma aproximação da função que associa x_i a y_i baseado em um grande número de pares (x_i, y_i) (conjunto de treinamento). Após o treinamento este conhecimento é utilizado para prever o valor correspondente $\hat{y}$ de uma entrada desconhecida x' (SARKAR; BALI; SHARMA, 2017). O

Figura 3 – Classificação dos algoritmos de aprendizado de máquina



Fonte – Sarkar, Bali e Sharma (2017)

aprendizado supervisionado ainda pode ser subdividido em duas categorias, dependendo do vetor de saídas desejadas y_i:

- **classificação**: nas tarefas de classificação a saída desejada pode assumir valores (reais ou inteiros) de um conjunto finito de categorias ou classes previamente estabelecidas (BISHOP, 2006). O problema de reconhecimento de dígitos escritos a mão, apresentado no início deste capítulo, é um exemplo de aplicação onde se utiliza o treinamento supervisionado do tipo classificação;
- **regressão**: quando a saída desejada é uma variável contínua e pode assumir qualquer valor real. Neste tipo de tarefa as variáveis de entrada ou atributos (elementos do vetor x_i) são denominadas de variáveis independentes, explicativas ou ainda preditoras, enquanto que a saída que se deseja estimar é denominada de variável dependente (SARKAR; BALI; SHARMA, 2017). Exemplos de treinamento supervisionado do tipo regressão incluem: a previsão do preço das ações de uma determinada empresa (TSAI; WANG, 2009) e aplicações em identificação de sistemas (CHIUSO; PILLONETTO, 2019).

b) **aprendizado não supervisionado**: neste tipo de treinamento os padrões de entrada x_i não são rotulados, ou seja, a saída desejada y_i não está disponível. As principais aplicações incluem:

- **clusterização**: os algoritmos de aprendizado de máquina nesta categoria

procuram descobrir padrões de similaridade entre os dados e agrupá-los em classes distintas entre si (*clusters*). Como os dados não são rotulados, cabe ao pesquisador atribuir significado a cada agrupamento (SARKAR; BALI; SHARMA, 2017). De posse do modelo treinado, novas observações podem ser classificadas nos *clusters* formados;

- **redução de dimensionalidade**: em algumas aplicações o vetor de entrada x_i pode apresentar um número muito elevado de atributos. Nesta situação, atividades como a análise e visualização dos dados podem se tornar bastante complicadas devido à complexidade do espaço dos atributos. Problemas quanto ao treinamento dos modelos e limitações quanto a memória e espaço de armazenamento também podem decorrer em função da alta dimensionalidade. Estes efeitos colaterais são denominados por "maldição da dimensionalidade" (SARKAR; BALI; SHARMA, 2017). A redução de dimensionalidade busca reduzir a quantidade de atributos mantendo as principais características da representação inicial (MOHRI; ROSTAMIZADEH; TALWALKAR, 2012). *Principal Component Analysis* (PCA) é um exemplo de algoritmo não supervisionado utilizado para a redução de dimensionalidade por um processo denominado de extração de atributos (SARKAR; BALI; SHARMA, 2017).

c) **aprendizado semi supervisionado**: neste cenário o algoritmo de aprendizado de máquina recebe uma grande quantidade de dados não rotulados e alguns exemplares rotulados. Este tipo de abordagem é utilizado, por exemplo, em situações onde o custo e tempo gastos na rotulagem dos dados são muito elevados tornando impeditivo a sua realização. No entanto, o objetivo do modelo treinado com aprendizado semi supervisionado também é o de classificação e regressão como ocorre no aprendizado supervisionado (CHAPELLE BERNHARD SCHöLKOPF, 2006);

d) **aprendizado por reforço**: ao contrário das três abordagens citadas anteriormente, no aprendizado por reforço, não está disponível para o algoritmo de aprendizado de máquina um conjunto de dados para treinamento. O aprendizado se dá pela interação com o ambiente que se deseja atuar por um determinado período com o objetivo de melhorar o desempenho de uma determinada tarefa.

O algoritmo inicia com um conjunto de políticas e estratégias para interagir com o ambiente. Após observar o ambiente ele escolhe uma ação baseada nas políticas e estratégias previamente definidas. Neste momento o algoritmo recebe uma resposta na forma de punição ou recompensa. Assim ele pode atualizar as suas políticas e estratégias, se necessário, de forma iterativa até que ele aprenda o suficiente sobre o ambiente para receber as respostas desejadas (SARKAR; BALI; SHARMA, 2017).

Dependendo do algoritmo de aprendizado de máquina, a forma de generalizar a partir dos dados de treinamento pode assumir basicamente dois modos:

a) **aprendizado baseado em instâncias**: os métodos enquadrados nesta categoria generalizam novos dados diretamente a partir do conjunto de treinamento armazenado em memória, ao invés de construírem explicitamente um modelo matemático. O algoritmo KNN se enquadra neste tipo de aprendizado. Nele, uma medida de similaridade (p.ex. distância Euclidiana) é utilizada para encontrar os representantes mais similares (o hiperparâmetro k define a quantidade) ao novo ponto de dados que se deseja generalizar. Caso a aplicação em questão se tratar de um problema de classificação, o novo dado será rotulado com a classe que pertente a maioria dos representantes mais próximos (SARKAR; BALI; SHARMA, 2017; BISHOP, 2006);

b) **aprendizado baseado em modelo**: é a abordagem mais tradicional de aprendizado de máquina (SARKAR; BALI; SHARMA, 2017). Aqui os dados do conjunto de treinamento são apresentados iterativamente ao algoritmo. Parâmetros internos ao modelo (também denominados de pesos) são ajustados de forma a otimizar determinada função objetivo.

O último aspecto de classificação diz respeito ao período em que o algoritmo recebe treinamento:

a) **aprendizado em lote**: também conhecido como aprendizado *offline*. Os algoritmos pertencentes a esta família, são treinados utilizando todos os dados disponíveis e ao atingirem o desempenho desejado são implantados em produção. No entanto, esses modelos não assimilam novas informações na medida que novos dados são disponibilizados. Ao longo do tempo este modelo pode se tornar

ineficiente sendo necessário novo treinamento. Nesta ocasião, o novo conjunto de treinamento será constituído pelos dados históricos (utilizados quando do primeiro treinamento) acrescidos dos novos dados disponíveis desde então. Esta abordagem pode representar um problema na medida em que o conjunto de treinamento se torna maior com o passar do tempo, consumindo mais processamento, memória e espaço de armazenamento (SARKAR; BALI; SHARMA, 2017);

b) **aprendizado *online***: da mesma forma que ocorre no treinamento em lote, o ajuste inicial acontece utilizando todos os dados disponíveis. No entanto, uma vez implantando em produção, os modelos com treinamento *online* continuam a aprender com novos dados na medida em que são disponibilizados, sem a necessidade de executar todo do treinamento com os dados históricos. Este tipo de treinamento também é denominado de aprendizado incremental ou continuado. Os dados históricos utilizados no treinamento continuado podem ser removidos uma vez que foram utilizados no ajuste incremental do modelo. Isto evita o problema de acúmulo de dados que estão sujeitos os modelos com treinamento em lote. No entanto, estes modelos podem ter o seu desempenho comprometido uma vez que estão suscetíveis a receberem amostras de dados ruins (SARKAR; BALI; SHARMA, 2017).

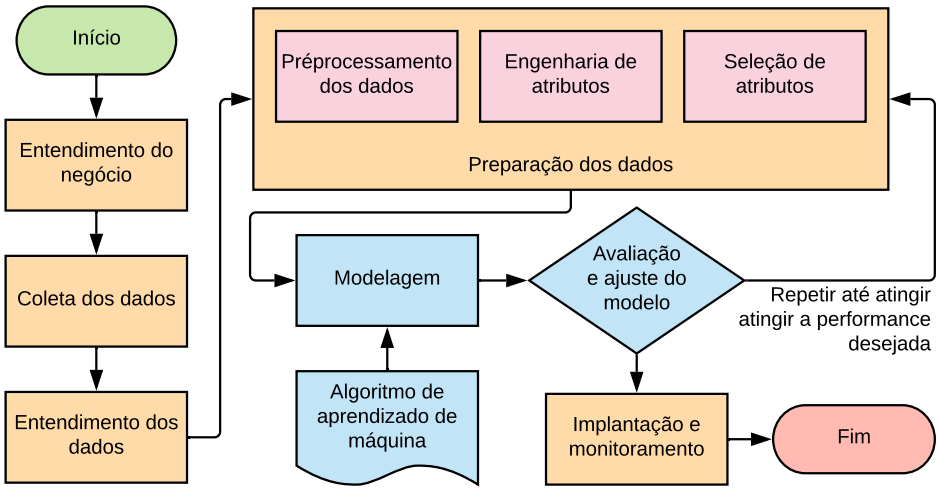
2.2 Fluxo de atividades do aprendizado de máquina

Em aplicações do mundo real, os dados coletados necessitam de tratamento apropriado antes de serem utilizado no treinamento dos modelos. Além disso, técnicas apropriadas devem ser empregadas de modo a garantirem o funcionamento correto dos mesmos e evitar problemas comuns em projetos de aprendizado de máquina. A adoção de um fluxo de trabalho pode ajudar nesse processo, orientando o pesquisador na escolha das melhores práticas.

A Figura 4 ilustra um típico fluxo de atividades em projetos de aprendizado de máquina. O mesmo é baseado na metodologia *Cross Industry Standard Process for Data Mining* (CRISP-DM) (CHAPMAN et al., 2000), que se tornou um processo de modelagem padrão da indústria em projetos de mineração de dados (SARKAR; BALI; SHARMA,

2017).

Figura 4 – Fluxo de atividades do processo de modelagem com aprendizado de máquina



Fonte – Adaptado de (CHAPMAN et al., 2000) e (SARKAR; BALI; SHARMA, 2017)

Nas próximas seções serão descritas as principais características de cada atividade do processo.

2.2.1 Entendimento do negócio

Antes de iniciar o processo de coleta e modelagem dos dados é necessário definir o problema que se pretende resolver em uma perspectiva de negócios. Entender o objetivo primário do cliente é fundamental para canalizar os esforços na direção correta. Outro fator igualmente importante é a definição do critério de sucesso. Isto pode ser feito através de uma métrica objetiva, como por exemplo, redução dos custos em 10% ou de forma subjetiva como "conseguir informações úteis sobre produtos relacionados" (CHAPMAN et al., 2000).

Na sequência deve-se analisar o contexto em que o problema está inserido, identificando recursos, restrições, hipóteses e outros fatores que poderão influenciar no processo de análise dos dados.

Uma vez identificado o escopo problema de negócio e o seu contexto, o analista deverá definir o problema de aprendizado de máquina correspondente. No exemplo de problema de negócio dado no início desta seção (redução dos custos em 10%), o problema de aprendizado de máquina poderia ser mapeado em "prever a quantidade de itens que serão adquiridos no próximo mês baseado em dados históricos de três anos, informações

demográficas e preço dos itens" (CHAPMAN et al., 2000). Da mesma forma, deverá ser definido o critério de sucesso para o problema de aprendizado de máquina, como, por exemplo, obter um certo nível de exatidão nas previsões.

2.2.2 Coleta dos dados

Nesta atividade serão coletados os dados das fontes identificadas no passo anterior. Pode ocorrer em paralelo com a etapa de entendimento do negócio, pois a disponibilidade e a qualidade dos dados podem influenciar na definição do problema. Aqui uma infinidade de fontes podem ser consideradas: *mainframes*, *websites*, aplicações *web*, banco de dados relacionais, arquivos de texto, sensores, dispositivos móveis, etc (SARKAR; BALI; SHARMA, 2017).

Dependendo da fonte, técnicas apropriadas devem ser utilizadas para a extração dos dados. Ao coletar informações de banco de dados relacionais, por exemplo, a linguagem *Structured Query Language* (SQL) é a ferramenta mais comum para a recuperação de dados. Se a fonte de dados considerada é um *website*, a técnica de *web scraping* pode ser a mais adequada (SARKAR; BALI; SHARMA, 2017).

O formato dos dados coletados também podem variar dependendo da fonte considerada. Dentre os mais comuns, podemos citar: *Comma-separated values* (CSV), *Java Script Object Notation* (JSON), *eXtensible Markup Language* (XML) e *Hyper Text Markup Language* (HTML). Cada formato exige um tratamento adequado de modo a se extrair as informações desejadas.

2.2.3 Entendimento dos dados

Esta atividade tem por objetivo melhorar o entendimento sobre os dados coletados e ainda avaliar a qualidade dos mesmos. Esta etapa do processo pode ser subdividida em três passos:

2.2.3.1 Descrição dos dados

É realizado um levantamento inicial dos dados que envolve a documentação: do volume dos dados (tamanho, quantidade de registros, tabelas); dos atributos (tipo de dados e descrição); relacionamento entre dados de diferentes fontes; estatísticas descritivas

básicas (média, mediana e variância) e identificação dos atributos mais importantes para a solução do problema (SARKAR; BALI; SHARMA, 2017).

2.2.3.2 Análise exploratória dos dados

Tukey (1977) compara a análise exploratória de dados com o trabalho de investigação criminal. Segundo o autor, o investigador, para obter sucesso no seu trabalho, deve possuir as ferramentas adequadas e entendimento sobre a área que está atuando. De forma semelhante, o cientista de dados deve possuir conhecimentos específicos sobre o problema que se pretende resolver. No entanto, na maioria das situações, o conhecimento aprofundado em uma determinada área pode representar um fator impeditivo à realização do projeto. Contudo, conhecimentos gerais sobre a análise de dados pode ser útil em várias situações.

O objetivo principal da análise exploratória é obter um entendimento mais detalhado sobre os dados. Para isso, o cientista de dados deverá utilizar ferramentas da estatística descritiva e recursos de visualização de dados. As principais tarefas desta atividade incluem (SARKAR; BALI; SHARMA, 2017):

a) explorar, descrever e visualizar os atributos dos dados;

b) selecionar atributos e subconjuntos de dados que parecem mais importantes;

c) análise de correlações e associações;

d) testes de hipótese;

e) identificação de dados ausentes.

2.2.3.3 Análise da qualidade dos dados

Nesta atividade os dados são analisados quanto à: identificação de valores ausentes; valores inconsistentes; informações incorretas devido a dados corrompidos e informações incorretas de metadados. Os resultados são documentados para que as intervenções apropriadas sejam realizadas nas atividades subsequentes (SARKAR; BALI; SHARMA, 2017).

2.2.4 Preparação dos dados

Em raras situações os dados coletados são utilizados na sua forma bruta pelos métodos de aprendizado de máquina. Nesta etapa os atributos passarão por um processo

de limpeza, correção e transformação denominado de pré-processamento ou *data wrangling*. Em muitos casos, mesmo depois do pré-processamento alguns atributos necessitam de uma etapa de transformação adicional derivando novos atributos mais apropriados para o treinamento dos modelos. Tal procedimento é realizado na atividade de engenharia de atributos. Finalmente os atributos passam por um processo de seleção onde será escolhido o conjunto que melhor representa o problema abordado.

2.2.4.1 Pré-processamento

Seguem nas próximas alíneas as principais tarefas executadas na atividade de pré-processamento (SARKAR; BALI; SHARMA, 2017):

- **filtragem dos dados**: seleção de um subconjunto de registros ou atributos. Ao finalizar a coleta dos dados, nem todos os atributos serão de interesse do cientista de dados. Da mesma forma pode ser feita uma seleção de registros baseado em um determinado critério como, por exemplo, pelo campo de data;
- **conversão de tipos**: conversão da informação para o tipo correto. Os campos que contêm data são exemplos típicos, pois, geralmente são coletados no formato de *string*. Para que seja possível realizar operações sobre este campo (filtragem, adição, etc), esta informação deve ser convertida para o tipo data na ferramenta que o cientista de dados esteja utilizando;
- **transformações**: nesta tarefa novos atributos são criados a partir da combinação ou não de atributos existentes. Por exemplo, em uma aplicação bancária, os clientes poderiam ser classificados dependendo do montante de aplicações que possuem. Ou ainda a extração do dia da semana de um campo de data;
- **preenchimento de valores ausentes**: a existência de valores ausentes nos atributos podem ocasionar diversos tipos de problemas se não forem corretamente tratados. A primeira abordagem é simplesmente a remoção dos registros que possuem atributos com valores ausentes. Esta prática é adotada quando se dispõe de um banco de dados significativamente grande. No entanto, em outras situações, estes valores devem ser preenchidos. A ação mais comum é o preenchimento dos valores ausentes com medidas de tendência central como a média ou a mediana. Em projetos com dados de séries

temporais, pode ser necessária a criação de novos registros para tornar a série igualmente espaçada.

2.2.4.2 Engenharia de atributos

De forma geral os algoritmos de aprendizado de máquina utilizam apenas informações numéricas para o treinamento dos modelos. Além disso, dependendo do algoritmo, a magnitude da informação também deve estar em uma determinada faixa. No entanto, os dados coletados podem se apresentar em diferentes formatos. Após a fase de pré-processamento, um tratamento especial ainda poderá ser necessário para alguns atributos. A atividade de derivar informações dos dados brutos, que sejam utilizáveis pelos algoritmos de aprendizado de máquina é denominado de engenharia de atributos (ZHENG; CASARI, 2018).

Considerada a etapa mais importante do fluxo de atividades do projeto de aprendizado de máquina, a atividade de engenharia de atributos influencia diretamente os resultados obtidos na fase de modelagem. Escolher os atributos mais adequados para o problema que se pretende resolver, pode ser a chave para se obter as respostas corretas (ZHENG; CASARI, 2018). Domingos (2012) enfatiza a importância dos atributos ao declarar que:

> Ao final do dia, alguns projetos de aprendizado de máquina serão bem-sucedidos e outros falharão. O que fez a diferença? Facilmente o fator mais importante são os atributos utilizados (DOMINGOS, 2012, tradução nossa)[3].

A engenharia de atributos é altamente dependente do problema em questão e exige a combinação de conhecimentos inerentes ao domínio do problema, experiência, intuição e ferramentas matemáticas para transformar dados brutos em atributos pertinentes (SARKAR; BALI; SHARMA, 2017). Em uma aplicação de reconhecimento de dígitos, por exemplo, a informação de tamanho do arquivo poderia ser irrelevante para o problema. Já o contorno da imagem pode representar um importante atributo. A lista abaixo exemplifica algumas possíveis informações que podem ser obtidas pela engenharia de atributos:

- idade de uma pessoa dado a data de nascimento e a data atual;

[3] *At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.* (DOMINGOS, 2012)

- média e mediana da quantidade diária de visualizações de um vídeo;
- quantidade de ocorrências de uma palavra em um documento;
- informações dos pixeis de uma imagem;
- desvio padrão do índice pluviométrico dos últimos dez dias.

Segundo Sarkar, Bali e Sharma (2017), ao utilizar a engenharia de atributos o cientista de dados pode se beneficiar dos seguintes aspectos:

a) **melhora a representação dos dados**: além de preparar os dados para o treinamento dos modelos de aprendizado de máquina, a engenharia de atributos torna os dados mais legíveis para o seu entendimento. Visualizar graficamente a frequência das palavras de um documento pode nos dar mais informações do que lidar com ele sem nenhum tipo de tratamento, por exemplo;

b) **melhora o desempenho dos modelos**: modelos treinados com os atributos corretos podem superar o desempenho de outros métodos mesmo que estes sejam mais sofisticados;

c) **etapa indispensável para a construção e avaliação dos modelos**: na etapa de construção e avaliação do modelo talvez seja necessário, iterativamente, retornar à atividade de engenharia de atributos até que se encontre o melhor conjunto de atributos para obter o melhor modelo;

d) **tipo de dados variados**: utilizar dados numéricos com alguma transformação é relativamente simples se comparado a tipos de dados como textos, vídeos e imagens. A engenharia de atributos pode ajudar na elaboração de modelos utilizando diversos tipos de dados, mesmo para tipos complexos e não estruturados;

e) **foco no problema do negócio**: a engenharia de recursos promove uma maior interação entre o cientista de dados e os especialistas do negócio. Esta comunicação é importante na medida em que se viabiliza a criação e identificação dos atributos que podem ser úteis para a solução do problema.

Dependendo do tipo de dado, técnicas específicas da engenharia de atributos devem sem empregadas para a obtenção da informação que viabiliza o treinamento dos modelos. A seguir, para cada tipo de dado, será feita uma breve abordagem das principais ferramentas.

A engenharia de atributos aplicada em imagens foge do escopo deste trabalho, portanto, não será abordada.

2.2.4.2.1 Engenharia de atributos para dados numéricos

Dados numéricos podem se originar de uma variedade de fontes: geolocalização de um lugar ou pessoa, valor de uma compra, medidas de um sensor, volume de tráfego de carros, etc. Apesar de estarem em um formato legível para os modelos de aprendizado de máquina, podem ser necessários alguns tipos de transformações.

Alguns aspectos sobre o dado numérico precisam ser analisados: a magnitude do dado, se valores negativos são permitidos e a granularidade dos dados. Outro fator importante é a escala do dado. Alguns algoritmos que utilizam distância Euclideana podem apresentar resultados distorcidos se os atributos estão em escalas muito diferentes, como é o caso do KNN (ZHENG; CASARI, 2018). A seguir será dada uma breve descrição sobre as técnicas aplicáveis a atributos numéricos:

a) **binarização**: às vezes estamos interessados representar apenas se houve ou não a ocorrência de um evento, ao invés de considerar a quantidade de ocorrências (ZHENG; CASARI, 2018). A Tabela 1 ilustra a criação do atributo **ouvida** derivado do atributo **quant_audicoes**. O novo atributo indica se a música foi ouvida pelo menos uma vês (indicado pelo valor um). Caso contrário o campo é preenchido com o valor zero.

Tabela 1 – Exemplo de binarização de atributos

id_usuario	id_musica	quant_audicoes	**ouvida**
1	451	0	0
2	851	2	1
3	624	10	1
4	628	11	1
5	863	0	0
6	624	20	1

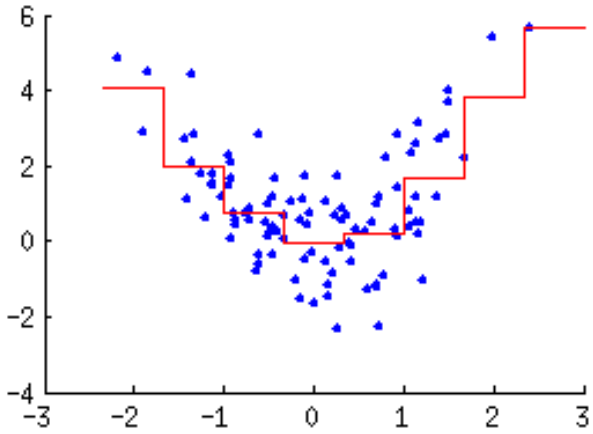
Fonte – Adaptado de (SARKAR; BALI; SHARMA, 2017)

b) **interações**: os atributos numéricos podem ser combinados com o objetivo de capturar possíveis relações entre si. Por exemplo os atributos x_1, x_2 e x_3 po-

dem ser combinados obtendo-se os novos atributos : $x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3$ e x_2x_3 (SARKAR; BALI; SHARMA, 2017);

c) **quantização**: nesta técnica valores numéricos contínuos são discretizados. São definidos intervalos de valores conhecidos como *bins* e então definido um valor correspondente para cada um deles. Cada observação do atributo que se deseja transformar assume o valor do *bin* ao qual ele pertence (está compreendido no intervalo). Esta técnica pode ser utilizada para minimizar os efeitos de dados ruidosos ou ainda para categorização. No entanto, a perda de informação é o efeito colateral mais importante dessa transformação. A Figura 5 ilustra a técnica de quantização (linha vermelha) aplicada ao conjunto de dados representados pelos pontos azuis.

Figura 5 – Exemplo de quantização de dados

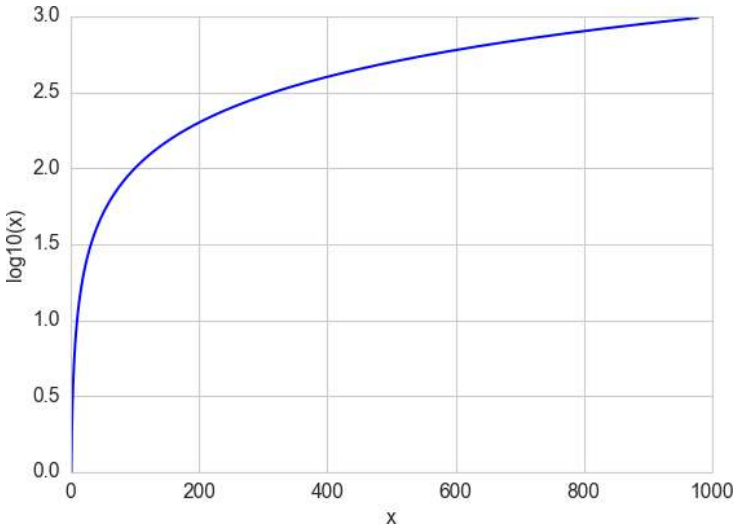


Fonte – (MINEAULT, 2013)

d) **transformação logarítmica**: a transformação logarítmica pode ser definida como $y = log_b(x)$. Na medida que ela expande valores que se concentram em magnitudes menores e comprime aqueles que estão em magnitudes maiores, ela é útil para corrigir distribuições que apresentam assimetria positiva dos dados. A Figura 6 evidencia o efeito da transformação log. Valores entre 100 e 1000 são mapeados no intervalo $[2, 3]$ (compressão), já valores entre zero e cem são expandidos para o intervalo $[0, 2]$.

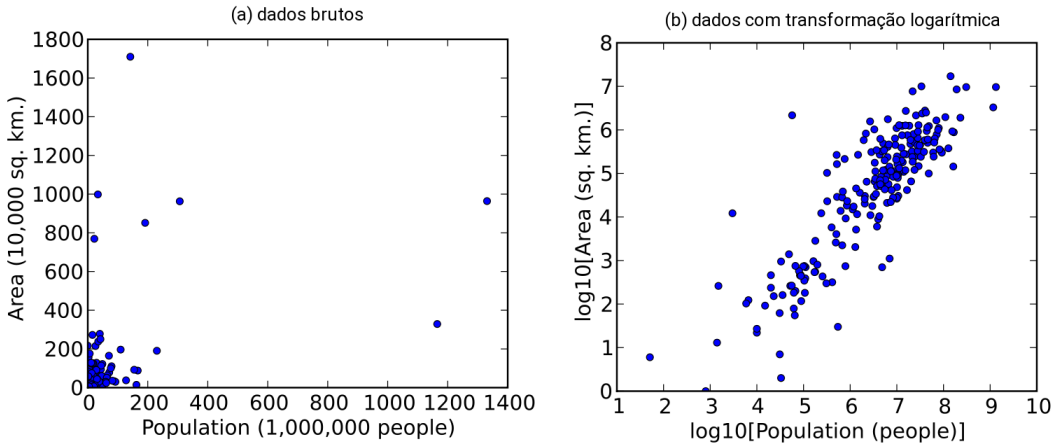
A Figura 7 apresenta uma aplicação prática da transformação logarítmica. Através do gráfico de dispersão fica claro como a relação entre as duas variáveis fica evidenciada após a aplicação da transformação.

Figura 6 – Exemplo de transformação log



Fonte – (ZHENG; CASARI, 2018)

Figura 7 – Aplicação da transformação logarítmica evidenciando a relação entre duas variáveis



Fonte – (Wikipedia contributors, 2018)

e) **normalização min-max**: em algumas situações são exigidos que os valores de um determinado atributo estejam dentro de um intervalo pré-estabelecido $[a, b]$. A normalização min-max atende a este requisito aplicando uma transformação linear nos dados originais, transformando-os para a escala desejada. Seja x a variável sobre a qual se deseja aplicar a normalização min-max, $min(x)$ e $max(x)$ os valores mínimo e máximo que x pode assumir. A normalização min-max é dada por:

$$x'_i = \frac{x_i - min(x)}{max(x) - min(x)}(b - a) + a \tag{2.1}$$

onde x'_i é o valor corresponde ao elemento x_i de x, transformado para o intervalo $[a, b]$ (ZHENG; CASARI, 2018).

f) **padronização**: frequentemente, os atributos de uma determinada observação variam em unidades de medidas e escala. Algoritmos que utilizam distância Euclidiana, por exemplo, podem gerar resultados distorcidos uma vez que atributos com as maiores magnitudes terão mais peso no cálculo da distância, se comparados a outros atributos de magnitudes menores. Em outros casos, o algoritmo pode assumir que os seus dados estejam em uma distribuição normal. Para superar estas questões, pode ser empregada a padronização dos atributos. Neste procedimento os dados são redistribuídos assumindo média zero e desvio padrão um. Seja x a variável cujos valores se deseja padronizar, $\overline{x}$ a média de x e σ_x o desvio padrão de x. O valor padronizado x'_i de um elemento $x_i \in x$ é dado por (ZHENG; CASARI, 2018):

$$x'_i = \frac{x_i - \overline{x}}{\sigma_x} \tag{2.2}$$

2.2.4.2.2 Engenharia de atributos para dados categóricos

Um atributo categórico representa um conjunto finito de categorias ou classes. As categorias podem se apresentar no formado numérico ou textual e são classificadas em variáveis nominais e ordinais. Nas variáveis categóricas nominais as classes não possuem uma relação de ordinalidade, como, por exemplo: gênero de filmes, nome de países, cores de objetos, etc. Já as variáveis categóricas ordinais podem ser ordenadas, por exemplo, dias da semana, tamanho de roupas e nível educacional (SARKAR; BALI; SHARMA, 2017).

Quando um atributo categórico se encontra no formato de texto, é necessário transformar cada categoria em uma representação numérica, já que os algoritmos de aprendizado de máquina não podem lidar diretamente com uma informação textual. Por exemplo, o atributo categórico "nível educacional" poderia ser mapeado para os seguintes valores inteiros: fundamental-0, ensino médio-1, graduação-2, especialização-3, mestrado-4 e doutorado-5. No entanto, este tipo de codificação pode inserir informações equivocadas nos modelos de aprendizado de máquina. Por exemplo, em uma aplicação que busca prever a evasão escolar, utilizando o nível educacional como um de seus atributos, a relação

equivocada "quanto maior o nível educacional, maior será a evasão" poderá ser assimilada pelo modelo uma vez que doutorado-5 > mestrado-4 > especialização-3 > graduação-2 > ensino médio-1 > fundamental-0. Nestes casos existem outras alternativas que podem ser utilizadas para a codificação de atributos categóricos. Nas próximas alíneas serão abordados dois métodos de codificação de atributos categóricos utilizados neste trabalho. No entanto, existem outros procedimentos disponíveis na literatura, cada um com suas vantagens e desvantagens (ZHENG; CASARI, 2018).

a) ***One-Hot Encoding***: dado um atributo categórico com k categorias, esta codificação transforma o atributo em um vetor de k elementos. Cada posição do vetor corresponde a uma categoria. A posição n assume o valor um indicando que a observação corresponde à n-ésima categoria ou zero caso contrário (SARKAR; BALI; SHARMA, 2017). No Quadro 2 abaixo, é apresentado um exemplo da codificação *one-hot encoding* aplicada ao atributo categórico dia da semana:

Quadro 2 – Atributo "dia da semana" codificado com *one hot encoding*

		one hot encoding						
Categoria	**Numérico**	**cat$_1$**	**cat$_2$**	**cat$_3$**	**cat$_4$**	**cat$_5$**	**cat$_6$**	**cat$_7$**
Segunda	0	1	0	0	0	0	0	0
Terça	1	0	1	0	0	0	0	0
Quarta	2	0	0	1	0	0	0	0
Quinta	3	0	0	0	1	0	0	0
Sexta	4	0	0	0	0	1	0	0
Sábado	5	0	0	0	0	0	1	0
Domingo	6	0	0	0	0	0	0	1

Fonte – autoria própria

b) ***Feature Hashing***: em cenários onde o atributo categórico pode assumir centenas ou milhares de categorias, a codificação *one-hot encoding* torna-se inviável, pelo alto custo computacional que ela pode acarretar. A codificação *feature hashing*, por sua vez, mapeia cada categoria original (de tamanho arbitrário) a um vetor de m posições, através da aplicação de uma função *hash*. No entanto, este procedimento pode provocar colisão ao mapear diferentes categorias a um mesmo vetor (ZHENG; CASARI, 2018).

2.2.4.2.3 Engenharia de atributos para textos

Se comparado aos atributos categóricos e numéricos, a engenharia de atributos para dados textuais pode ser mais desafiadora. Por se tratar de um dado não estruturado, este tipo de atributo pode assumir diversos tipos de sintaxes, formatos e conteúdos. Além disso, é necessário traduzir a informação textual para uma representação numérica, para que ela possa ser entendida pelos modelos de aprendizado de máquina (SARKAR; BALI; SHARMA, 2017).

De forma semelhante ao dado numérico, o atributo textual também pode conter informações classificadas como ruídos (ZHENG; CASARI, 2018). O primeiro passo, antes de construir uma representação numérica para o atributo textual, é submete-lo a um pré-processamento. A seguir serão apresentadas, de forma sucinta, as principais técnicas de pré-processamento:

a) ***tokenização***: é a ação de decomposição do documento (qualquer representação textual, independente do seu tamanho) em componentes menores que possuem uma sintaxe e semântica definida. Por exemplo, um parágrafo possui diversos componentes incluindo sentenças que podem ser quebradas em orações que por sua vês pode ser quebrada em frases e palavras. O processo de tokenização é realizado utilizando delimitadores especiais tais como: ponto final, espaço em branco, quebras de linhas, tabulação, etc. A estes componentes extraídos dos documentos dá-se o nome de *tokens* (ZHENG; CASARI, 2018).

b) **remoção de caracteres especiais**: consiste em remover caracteres que não possuem significado importante para o processo de engenharia de atributos, por exemplo: ?, $, &, *, %, @, (.

c) **expansão de contrações**: contrações são representações reduzidas de palavras. São muito comuns, por exemplo, na língua inglesa. Este tipo de construção pode representar problemas no processo de engenharia de atributos para textos, dado que uma mesma palavra pode ser representada, por no mínimo, duas maneiras diferentes. Além de que, as contrações em Inglês, possuem o caractere especial apóstrofo, que deverá ser removido (ZHENG; CASARI, 2018).

d) **remoção das *stopwords*** : remoção de palavras que possuem pouca ou nenhuma importância quanto ao seu significado dentro do documento. Exemplos:

de, a, o, que, e, do, etc.

e) **correção de erros ortográficos**: compreende a correção de palavras que estão escritas incorretamente.

f) **padronização de caixa das letras**: a padronização de caixa das letras (maiúsculas ou minúsculas) simplifica procedimentos como, por exemplo a combinação de palavras ou *tokens*.

g) ***stemming***: processo de redução de palavras flexionadas ou derivadas ao seu radical. Por exemplo, as palavras abalada, abalado, abalaram, abalos e abalou possuem o mesmo radical: "abal". Tal procedimento trata as palavras de mesmo radical como sinônimos, dando a mesma importância para elas dentro do documento (ZHENG; CASARI, 2018).

h) **lematização**: redução da palavra à sua forma equivalente no gênero masculino e singular, também chamada de lema. Por exemplo, as palavras "gato", "gata", "gatos" e "gatas" são todas formas do mesmo lema: "gato". Para os verbos, o seu lema assume a forma nominal no infinitivo.

Após o procedimento de pré-processamento do atributo textual, o próximo passo consistem em construir uma representação numérica para cada documento. Uma abordagem comumente utilizada é o Modelo de Espaço Vetorial (MEV), onde cada documento é representado por um vetor, definido a seguir: seja um *corpus* documental C contendo N documentos e i termos (*tokens*). Seja D_j um documento de C. D_j é representado vetorialmente como:

$$D_j = \{w_{1,j}, w_{2,j}, w_{3,j}, \ldots, w_{i,j}\} \tag{2.3}$$

onde $w_{i,j}$ é o peso do i-ésimo *token* de C relativo ao documento D_j (ZHENG; CASARI, 2018). Dentre os diversos métodos para cálculo dos pesos, será abordado a seguir o modelo *Term Frequency-Inverse Document Frequency* (TF-IDF), devido a sua utilização no experimento deste trabalho. Desta forma, o peso $w_{j,i}$ é calculado de acordo com a equação:

$$w_{i,j} = tf\text{-}idf_{i,j} = tf_{i,j} \times idf_i \tag{2.4}$$

O termo $tf_{i,j}$ é dado pela quantidade de vezes que o *token* i ocorre no documento

j. Esta métrica também é conhecida como *bag of words*. O termo idf_i é dado por:

$$idf_i = \log \frac{N}{df_i} \tag{2.5}$$

onde N é a quantidade de documentos do *corpus* e df_i é o número de documentos em que o *token* i ocorre. Assim o peso $w_{i,j}$ dado ao termo i dentro do documento j é (MANNING; RAGHAVAN; SCHUTZE, 2008):

a) maior quando i ocorre muitas vezes dentro de um número pequeno de documentos, indicando que o termo é altamente discriminante para estes documentos;

b) menor quando i ocorre poucas vezes em um documento ou quando ocorre em muito documentos, indicando que o termo representa pouca importância;

c) menor quando i ocorre em praticamente todos os documentos, indicando que o mesmo é muito comum, portanto, possui pouca capacidade de discriminação.

2.2.4.3 Seleção de atributos

Ao final da etapa de engenharia de atributos o cientista de dados pode obter centenas ou milhares de atributos. No entanto, trabalhar com uma grande quantidade de atributos tornam os modelos muito complexos e pode levar à condição de sobreajuste. Nesta situação o modelo se ajusta demasiadamente ao conjunto de treinamento e a sua capacidade de generalização é comprometida. O objetivo desta etapa é encontrar o melhor conjunto de atributos que maximizam o desempenho dos modelos (SARKAR; BALI; SHARMA, 2017).

Os métodos de seleção de atributos se subdividem basicamente em três categorias:

a) **métodos de filtro**: utilizam técnicas de seleção de atributos independente do modelo de aprendizado de máquina que se pretende utilizar. Pode ser considerado como um tipo de procedimento de pré-processamento. Consideram apenas informações inerentes aos atributos e variáveis de saída (STAńCZYK, 2015). Um exemplo de seleção por filtro é a remoção de atributos que possuem variância abaixo de um limiar pré-estabelecido (SARKAR; BALI; SHARMA, 2017). Neste trabalho foi utilizado o coeficiente de correlação de Pearson (MUKAKA, 2012)

(equação 2.6) para eliminar atributos altamente correlacionados.

$$\rho = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} = \frac{\text{cov}(X, Y)}{\sqrt{\text{var}(X) \cdot \text{var}(Y)}} \tag{2.6}$$

b) **métodos *wrappers***: nesta abordagem o estimador utilizado é ajustado e avaliado para todos os possíveis subconjuntos de atributos. É garantido que se encontre a melhor combinação de atributos, porém, possui um alto custo computacional (STAńCZYK, 2015).

c) **métodos *embedded***: alguns algoritmos de aprendizado de máquina possuem embutido no processo de treinamento, seus próprios métodos para a seleção de atributos. Exemplo de estimadores nesta categoria estão os algoritmos Árvores de Decisão e Floresta Aleatória. Estes estimadores, no processo de treinamento, atribuem uma medida de importância aos atributos. De forma iterativa os atributos de menor importância podem ser removidos, e o conjunto resultante reavaliado. O processo se repete até que se encontre um conjunto com a quantidade de atributos desejada. Esta técnica é denominada de eliminação recursiva de atributos (SARKAR; BALI; SHARMA, 2017).

2.2.5 Avaliação de modelos e otimização de hiperparâmetros

Uma vez que os dados se encontram em um formato adequado para os algoritmos de aprendizado de máquina, esta etapa do processo aborda as práticas necessárias para se obter modelos bem ajustados e com boa capacidade de generalização. Inicialmente faz-se necessário a definição de métricas que possibilitam quantificar e comparar o desempenho dos modelos. Posteriormente será abordada uma estratégia de ajuste fino dos hiperparâmetros denominada de busca aleatória, combinada com o procedimento de validação cruzada.

2.2.5.1 Métricas de avaliação para modelos de regressão

Dependendo to tipo do modelo utilizado (classificação, regressão ou clusterização) deve-se adotar métricas específicas para avaliação do seu desempenho. Nesta seção serão abordadas duas métricas aplicáveis a modelos de regressão: *Mean Absolute Error* (MAE) e RMSE.

Considere um conjunto de n observações da variável dependente y, representados por $\{y_1, y_2, \ldots, y_n\}$. Deseja-se avaliar a acurácia das respostas de um modelo M qualquer, utilizando como referência as observações de y. Sejam $\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n\}$ as respostas do modelo M correspondentes a cada uma das n observações da variável y.

A primeira métrica, denominada MAE é definida pela equação (2.7) (HYNDMAN; ATHANASOPOULOS, 2018). Ela calcula, em média, o quanto o modelo errou, independentemente se foi para mais ou para menos.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.7}$$

Outra medida de acurácia muito utilizada é a RMSE, dada pela equação (2.8).

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{2.8}$$

Apesar de fácil interpretação, a métrica MAE dá o mesmo peso a todos os erros. Por outro lado, RMSE tende a penalizar modelos que cometem erros grandes devido a expressão quadrática em sua equação. Ambas as métricas possuem a mesma unidade de medida da variável de interesse, portanto, não podem ser utilizada para comparar o desempenho de modelos ajustados com diferentes variáveis (ADHIKARI; AGRAWAL, 2013).

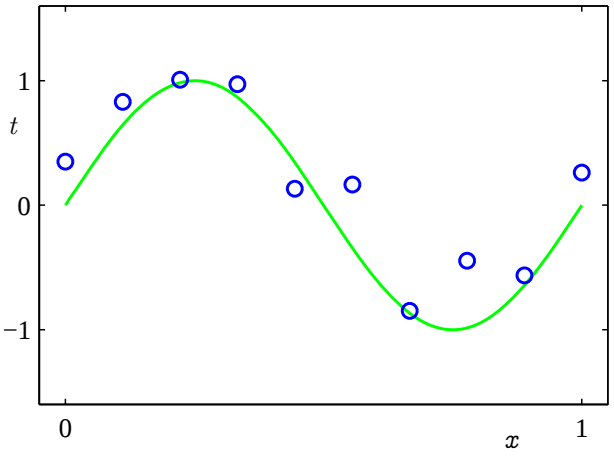
2.2.5.2 Otimização do modelo

Uma das atividades da etapa de pré-processamento é a divisão dos dados em conjunto de treinamento e testes. Como visto, o conjunto de treinamento é utilizado para o ajuste dos parâmetros internos do modelo, através do procedimento de treinamento, e o conjunto de testes é empregado para avaliar a sua capacidade de generalização frente a dados desconhecidos. Ao construir um modelo utilizando aprendizado de máquina, deseja-se que o mesmo absorva as informações relevantes contidas no conjunto de treinamento, e que, ao mesmo tempo, apresente a melhor capacidade de generalização possível. No entanto, este objetivo constitui um importante problema de aprendizado de máquina denominado de dilema viés-variância ou *bias-variance trade-off* (BISHOP, 2006).

Para melhor entendimento do dilema viés-variância, considere um problema hipotético onde, dada uma variável de entrada x, desejamos generalizar a variável de saída t correspondente. Os dados para este exemplo serão gerados pela função $f(x) = seno(2\pi x)+r$, onde r representa um ruído aleatório de distribuição Gaussiana (BISHOP, 2006, p.23).

Considere o conjunto de dados $\mathbf{X} \equiv (x_1, x_2, \ldots, x_N)^T$ formado por N observações de x, juntamente as observações da variável de saída denotada por $\mathbf{t} \equiv (t_1, t_2, \ldots, t_N)^T$, onde $t_n = f(x_n)$. A Figura 8 representa graficamente um conjunto de dez pontos (círculos azuis) obtidos escolhendo-se x igualmente espaçados no intervalo $[0, 1]$.

Figura 8 – Conjunto de pontos obtidos utilizando a função $f(x) = seno(2\pi x) + r$, onde r representa um ruído Gaussiano. A linha verde representa a função $seno(2\pi x)$ sem a presença de ruídos.



Fonte – Bishop (2006, p.23)

Estamos interessados em criar um modelo que a partir desse conjunto de dados possa generalizar o valor $\hat{y}$ para uma entrada $\hat{x}$ desconhecida. Implicitamente, estamos tentando descobrir a função $seno(2\pi x)$ que deu origem aos dados. Para este problema hipotético, considere o método de ajuste de curva utilizando a função polinomial dada pela equação (2.9).

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j \tag{2.9}$$

onde M é a ordem do polinômio. Os coeficientes $w_0, w_1, \ldots, w_M$ são denotados pelo vetor $\mathbf{w}$.

Os valores dos coeficientes serão determinados pelo ajuste do polinômio aos dados de treinamento. Isto pode ser feito minimizando a função de erro definida pela equação (2.10), cuja solução será denotada pelo vetor $\mathbf{w}^*$.

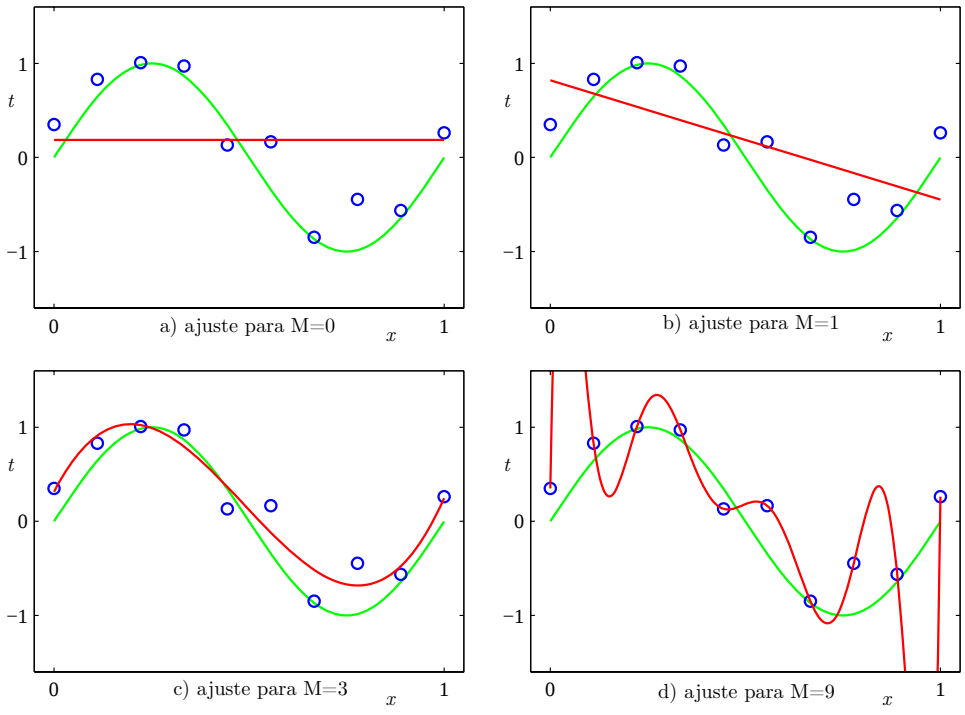
$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\} \tag{2.10}$$

No entanto, ainda temos que escolher a ordem M do polinômio. No campo de aprendizado de máquina, a ordem do polinômio pode ser entendida como sendo o hiperparâmetro do modelo. Esta escolha representa uma importante decisão no processo e

determinará a qualidade de generalização do modelo. Este é o foco principal desta seção: a otimização de modelos pelo ajuste dos hiperparâmetros.

Retornando ao exemplo proposto, considere as seguintes escolhas para a ordem do polinômio: $M = 0, 1, 3$ e 9. Os resultados dos ajustes de curva para cada caso, estão representados graficamente na Figura 9.

Figura 9 – Representação gráfica do ajuste de curva do polinômio para as ordens $M = 0, 1, 3$ e 9



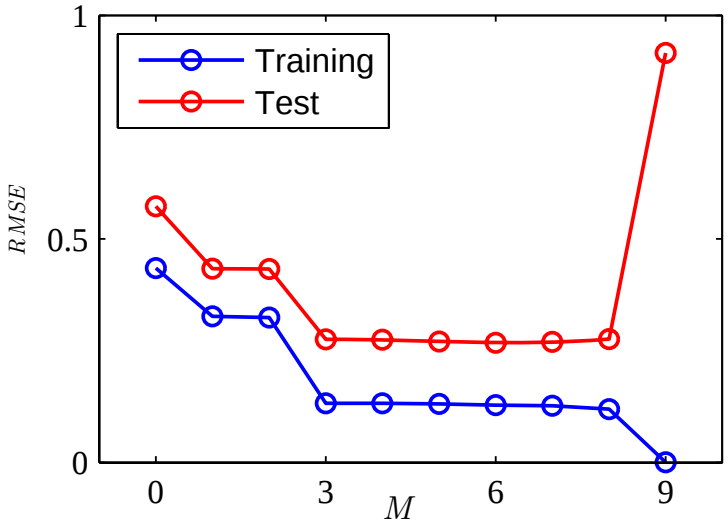
Fonte – Bishop (2006, p.26)

Podemos observar que para $M = 0$ e 1 foi obtido um péssimo ajuste aos dados de treinamento e consequentemente uma representação ruim da função $seno(2\pi x)$. Por outro lado, o ajuste do polinômio de ordem $M = 3$ parece ser o que mais se aproxima da função senoidal. Para $M = 9$ conseguimos um excelente ajuste aos dados de treinamento, pois a curva passa por todos os pontos obtendo então $E(\mathbf{w}^*) = 0$. No entanto, a curva resultante possui várias oscilações, não representando corretamente a função $seno(2\pi x)$. Esta última situação é conhecida como sobreajuste ou *overfitting*.

Como exposto no início da seção, o objetivo da modelagem com aprendizado de máquina é a construção de modelos que possuem ótima capacidade de generalização frente a novos dados. Podemos então quantificar a capacidade de generalização do polinômio para cada uma das escolhas da ordem M. Para isso foi gerado um conjunto de testes

contendo 100 observações utilizando a mesma função $f(x)$ empregada para a construção do conjunto de treinamento. Para cada escolha de M foi avaliado o erro cometido no conjunto de treinamento e no conjunto de testes através da métrica RMSE. O resultado do procedimento está representado graficamente na Figura 10.

Figura 10 – Representação gráfica da métrica RMSE para o conjunto de treinamento e teste para diversos valores de M.



Fonte – Bishop (2006, p.27)

Nota-se que para valores de M no intervalo $[0, 3[$ o resultados da métrica RMSE apresentam valores relativamente ruins devido à incapacidade que o polinômio possui de capturar as oscilações da função $seno(2\pi x)$. Para M no intervalo $[3, 8]$, encontramos os melhores resultados para o valor do erro tanto no conjunto de treinamento quanto no conjunto de testes. Tal fato pode ser comprovado examinando a Figura 9c que apresenta o melhor ajuste para $M = 3$. Para $M = 9$ o erro no conjunto de treinamento é reduzido a zero, no entanto, é muito elevado para o conjunto de teste, comprometendo completamente a capacidade de generalização do modelo.

De forma geral o erro de generalização pode ser definido pela seguinte equação:

$$\text{Erro de generalização} = \text{Erro de viés} + \text{Erro de variância} + \text{Erro irredutível} \quad (2.11)$$

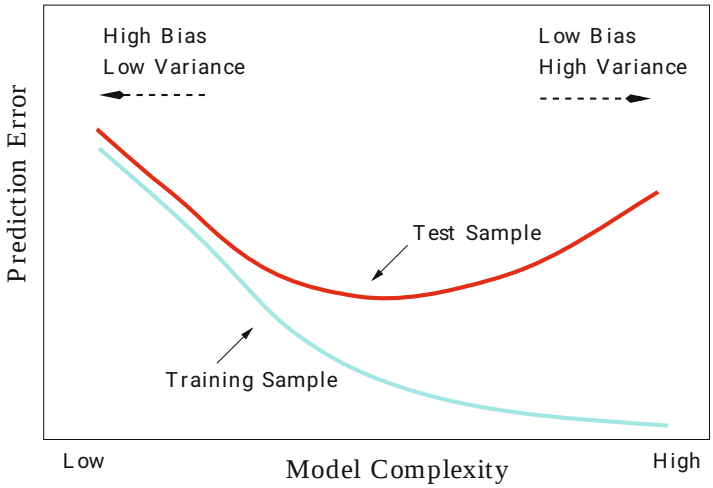
O erro de viés pode ser definido como a diferença entre o valor previsto pelo modelo e o valor real da variável que estamos tentando prever. É provocado devido a suposições equivocadas do modelo frente ao conjunto de treinamento utilizado. Por exemplo, utilizar uma regressão linear em um conjunto de treinamento cuja variável de saída tem relação

quadrática com as variáveis de entrada por resultar em um erro de viés elevado (SARKAR; BALI; SHARMA, 2017).

O erro de variância é definido como a sensibilidade que o modelo possui em detrimento às alterações no conjunto de treinamento. Um modelo bem condicionado consegue captar a relação entre as variáveis de saída e entrada independente da escolha dentre os possíveis conjuntos de treinamento que podem ser obtidos pela amostragem das variáveis de interesse. Um modelo sobreajustado, apresentará muitas flutuações nas respostas frente a novos padrões de entrada (alta variância) em decorrência da mudança do conjunto de treinamento. Por outro lado, o modelo subajustado terá a tendência de apresentar respostas mais ou menos constantes independentes dos padrões de entrada (baixa variância) (SARKAR; BALI; SHARMA, 2017).

De forma geral a variância tende a crescer com o aumento da complexidade do modelo e, ao mesmo tempo, o viés tende a diminuir. No entanto, a capacidade de generalização fica comprometida (erro elevado no conjunto de testes). Esta relação está ilustrada na Figura 11.

Figura 11 – Erro de treinamento e teste em função da complexibilidade do modelo



Fonte – Hastie Robert Tibshirani (2009, p.38)

Com a exposição deste exemplo hipotético, fica claro identificar o problema encontrado no dilema viés-variância: o modelo deve ser bem ajustado ao conjunto de treinamento, mas sem perder a sua capacidade de generalização. O desafio consiste então em atingir um equilíbrio entre os dois objetivos. Nas próximas seções serão abordadas práticas que auxiliam o pesquisador a obter modelos bem ajustados.

2.2.5.2.1 Hiperparâmetros

Uma das formas de controlar a complexidade de um modelo de aprendizado de máquina é através do ajuste de hiperparâmetros. Como visto, este aspecto influencia diretamente no balanço viés-variância. Os hiperparâmetros podem ser definidos como "meta parâmetros" do modelo e são definidos antes que se inicie o seu treinamento (SARKAR; BALI; SHARMA, 2017).

Cada algoritmo de aprendizado de máquina possuem seus próprios conjunto de hiperparâmetros. A exemplo do que foi abordado na seção 2.2.5.2, o parâmetro M do polinômio pode ser considerado como um hiperparâmetro.

Na próxima seção será abordado um método de avaliação de modelos denominado de Validação Cruzada (VC). O mesmo será utilizado posteriormente no procedimento de otimização de hiperparâmetros.

2.2.5.2.2 Validação Cruzada

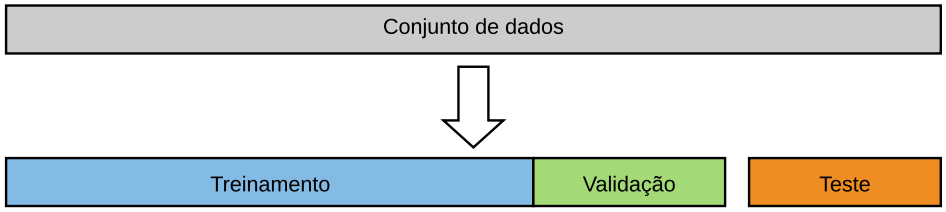
Como visto, a acurácia de um modelo pode ser medida comparando as suas respostas com um conjunto de dados não utilizados para o treinamento. Este procedimento pode ser empregado no processo de ajuste de hiperparâmetros utilizando a técnica de busca em grade ou *grid search*. Em uma abordagem mais ingênua, o conjunto de dados pode ser dividido em treinamento e teste. Logo em seguida, diversos modelos candidatos[4] são ajustados utilizando o conjunto de treinamento e avaliado no conjunto de teste, e então aquele que apresentar o melhor resultado é escolhido e finalmente sua capacidade de generalização é avaliada no conjunto de teste.

No entanto, nesta abordagem ocorre uma situação denominada de vazamento de dados. No caso, o conjunto de testes está sendo utilizado para estimar os hiperparâmetros do modelo e logo após utilizado para avaliá-lo, gerando certamente um resultado tendencioso.

Em uma segunda abordagem mais elaborada, o montante de dados disponíveis são divididos em três conjuntos: treinamento, validação e teste, conforme ilustrado na Figura 12. Aqui os modelos candidatos são ajustados com o conjunto de treinamento, mas a sua avaliação ocorre no conjunto de validação, escolhendo-se então aquele que apresentar melhores resultados.

[4] Um modelo por ser diferente de outro simplesmente pela variação de hiperparâmetros, mesmo quando utilizam o mesmo algoritmo.

Figura 12 – Divisão dos dados nos conjuntos de treinamento, validação e teste

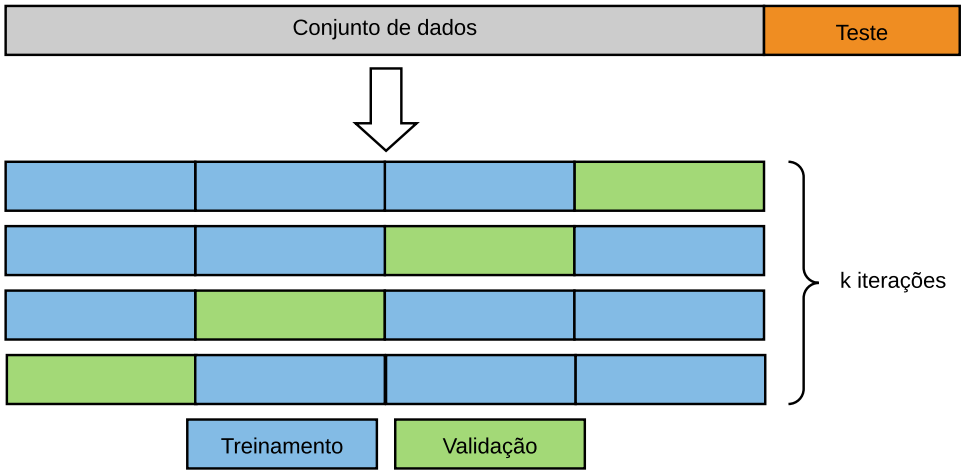


Fonte – autoria própria

Este procedimento também pode apresentar problemas, conduzindo o modelo escolhido para uma situação de sobreajuste se o conjunto de validação não for suficientemente representativo (BISHOP, 2006). Uma solução mais robusta que pode ser utilizada nesta atividade de otimização de modelos é a VC.

Na VC, uma parcela dos dados é reservada para o conjunto de testes, como ocorre nas outras situações. O restante dos dados são divididos em k conjuntos iguais. O modelo será treinado e avaliado k vezes, sendo que a cada iteração, $k-1$ conjuntos são utilizados para o treinamento e o conjunto restante utilizado para avaliação. O resultado final será obtido pela média aritmética das avaliações parciais. A Figura 13 ilustra graficamente o procedimento descrito. Este método de validação cruzada é denominado de *k-fold* (BISHOP, 2006, p.32).

Figura 13 – Representação gráfica do procedimento de validação cruzada



Fonte – autoria própria

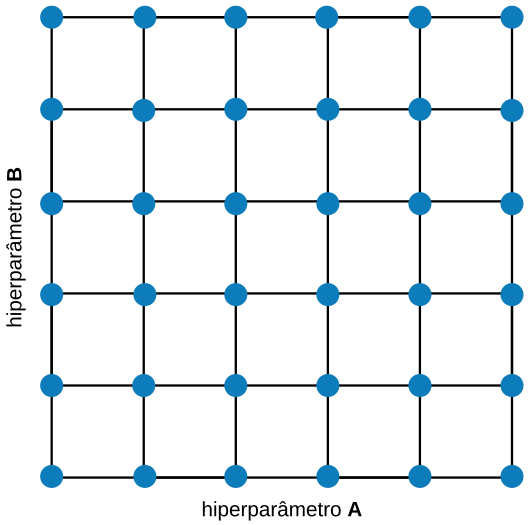
O maior problema da VC é o custo computacional. Alguns problemas, por si só, demandam um alto poder computacional para serem modelados. Dependendo do tamanho

de k e a quantidade de hiperparâmetros que se deseja otimizar, tal procedimento pode se tornar inviável (BISHOP, 2006).

2.2.5.2.3 Otimização de hiperparâmetros com busca aleatória

A primeira e mais simples estratégia para a otimização de hiperparâmetros é chamada de busca em grade ou *grid search*. Define-se inicialmente uma lista de valores que cada hiperparâmetro pode assumir dentro do espaço de busca (por exemplo, n valores igualmente espaçados). Então, para cada combinação possível de valores (Figura 14), é criado um modelo candidato, e sua acurácia é avaliada em regime de validação cruzada. Aquele que apresentar melhor resultado, será escolhido (SARKAR; BALI; SHARMA, 2017).

Figura 14 – Exemplo de busca em grade para dois hiperparâmetros. Os pontos em azul representam as possíveis combinações de valores.



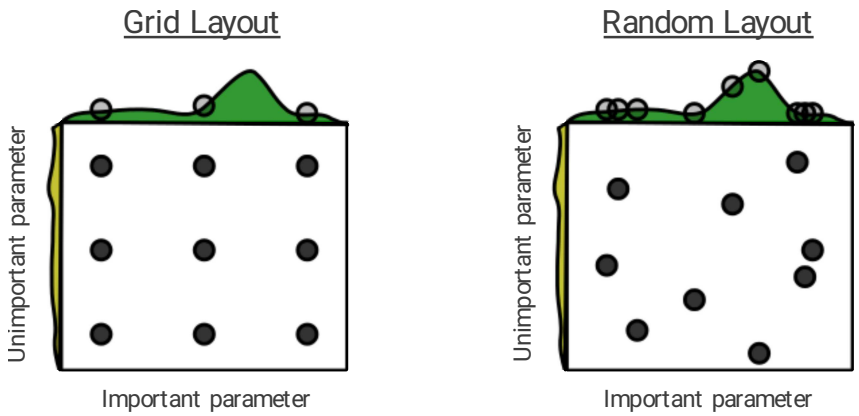
Fonte – autoria própria

A principal desvantagem do método é o custo computacional. Associado com a VC e a complexibilidade do problema, esta abordagem pode demandar muito tempo para ser executada ou até mesmo ser inviabilizada. No entanto, ela pode ser facilmente paralelizada. Outra desvantagem é que a melhor solução pode não estar entre as possíveis combinações de valores, uma vez que o espaço de busca é definido pelo usuário (SARKAR; BALI; SHARMA, 2017).

Uma alternativa à busca em grade é a busca aleatória. Esta abordagem, através de resultados empíricos, apresentou resultados iguais ou superiores à busca em grade, demandando menos tempo de execução (BERGSTRA; BENGIO, 2012). Neste procedimento, ao invés de informar uma lista de valores para cada hiperparâmetro, os mesmos são obtidos aleatoriamente a partir de uma distribuição uniforme, dentro de um espaço de busca delimitado. A cada iteração um modelo candidato é criado e avaliado em regime de validação cruzada, escolhendo-se o que apresentar melhor acurácia (SARKAR; BALI; SHARMA, 2017).

Intuitivamente, a Figura 15 explica porque o método de busca aleatória é mais eficiente. Nela estão representados a aplicação dos dois métodos para a otimização de um modelo com dois hiperparâmetros, sendo que um deles possui maior importância que o outro. Na figura da esquerda (busca em grade) o espaço de busca é uniformemente mapeado. No entanto, a projeção dos pontos na dimensão mais importante produz uma cobertura ineficiente. Por outro lado, na busca aleatória, os mesmos nove pontos estão distribuídos de maneira um pouco menos uniforme pelo espaço de busca, mas a projeção na dimensão da variável importante consegue uma melhor distribuição (BERGSTRA; BENGIO, 2012).

Figura 15 – Comparativo entre os métodos de otimização de hiperparâmetros: busca em grade e busca aleatória.



Fonte – Bergstra e Bengio (2012)

3 REDES NEURAIS ARTIFICIAIS ART

O ser humano aprende gradualmente frente a uma sucessão de eventos que ocorrem em tempo real. Ainda que as regras do ambiente no qual está inserido mudem, podemos nos adaptar novamente, mesmo que não tenhamos uma orientação externa para tal. Além disso, podemos aprender rapidamente novos fatos sem que, forçosamente, tenhamos que esquecer o que já sabemos (GROSSBERG, 2013). Inspirado pela capacidade que o cérebro biológico possui de aprender a categorizar, reconhecer e prever objetos e eventos em um ambiente em constante mudança, Grossberg (1976a), Grossberg (1976b) estabelece uma teoria que propõe explicar como o cérebro humano processa as informações cognitivas, denominada ART.

Existem na literatura outros modelos de redes neurais que buscam emular esta capacidade de aprendizagem dos seres humanos tais como: mapas auto-organizáveis, *back propagation*, recozimento simulado, *neocognitron*, máquinas de vetores de suporte, regularização e Modelos Bayesianos. No entanto, estes modelos são acometidos por um efeito indesejado denominado esquecimento catastrófico (GROSSBERG, 2013). Tal anomalia ocorre devido à incapacidade que a rede neural tem de lidar com o dilema plasticidade-estabilidade (GROSSBERG, 1982): a rede neural deve ter a aptidão de assimilar novas informações frente a um ambiente em constante mudança (plasticidade) e, ao mesmo tempo, preservar a integridade dos conhecimentos adquiridos anteriormente (estabilidade). A principal contribuição da ART está em resolver o dilema plasticidade-estabilidade (GROSSBERG, 1982).

Embora que, a teoria de Grossberg esteja fundamentada na neurobiologia, psicologia e em uma rigorosa representação matemática, este trabalho limita-se ao estudo de algoritmos que incorporam alguns princípios fundamentais da teoria ART. Na próxima seção será abordada a arquitetura básica da rede neural ART que dará origem a modelos mais sofisticados.

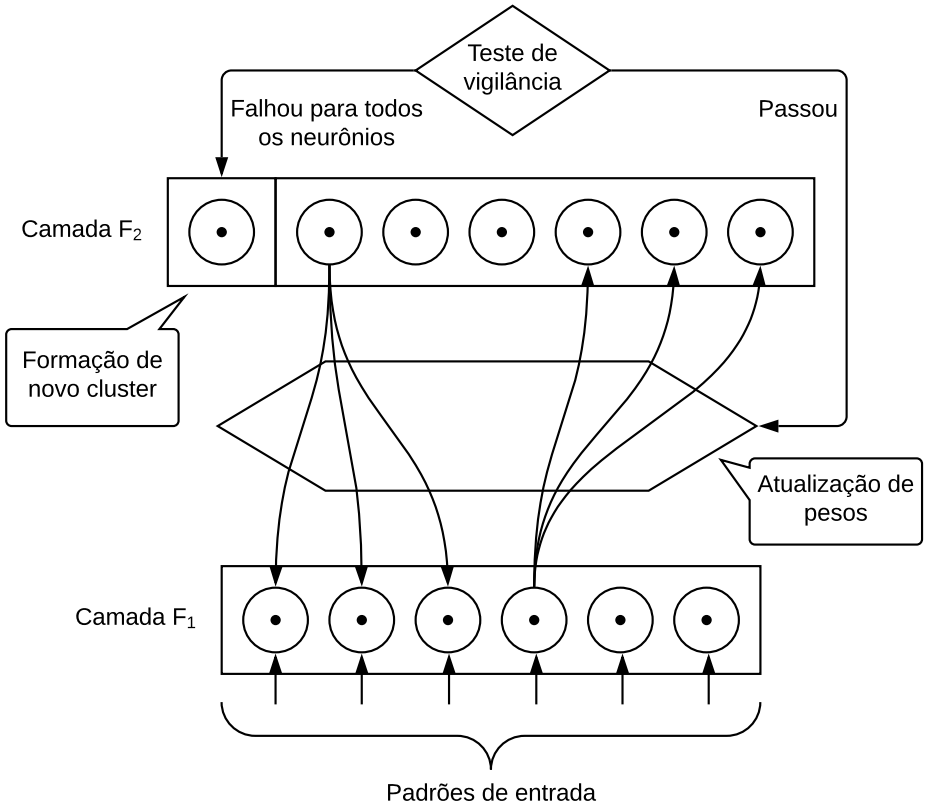
3.1 Arquitetura simplificada

ART é uma rede neural auto-organizável de treinamento não supervisionado com a capacidade de categorização e reconhecimento de padrões. Possui um mecanismo que permite, de forma adaptativa, a expansão da camada de saída adicionando novos neurônios

até que a quantidade apropriada seja atingida, dependendo dos dados de entrada. Este crescimento é regulado através do teste de vigilância que determina se uma entrada é suficientemente diferente das categorias existentes. Os neurônios da camada de saída representam as categorias ou *cluster*, nos quais os padrões de entrada vão sendo acomodados. Cada neurônio é constituído por um vetor de pesos denominado de protótipo da categoria, que representa todas as entradas que ali foram associadas (RAJASEKARAN; PAI, 2017).

A rede neural ART é constituída por três elementos básicos: camada de entrada F_1, camada de saída F_2 e um mecanismo de controle de similaridade de padrões (Figura 16). A camada F_1 é completamente conectada à camada F_2, ou seja, cada neurônio da camada F_1 é conectado a todos os neurônios da camada F_2 através de ligações ponderadas por pesos, denominados de pesos *bottom-up*. Da mesma forma a camada F_2 é completamente conectada à camada F_1 pelos pesos *top-down* (RAJASEKARAN; PAI, 2017).

Figura 16 – Arquitetura simplificada da rede neural ART



Fonte – Rajasekaran e Pai (2017)

Apresentado um padrão na entrada da rede, os neurônios da camada F_2 recebem estímulos ponderados pelos pesos *bottom-up*. Dessa forma os neurônios competem entre si

segundo o princípio "ganhador leva tudo". Aquele que receber o sinal de maior intensidade vence a competição. Sendo este considerado como uma hipótese, será avaliado no teste de vigilância antes de aprender o padrão de entrada (CARPENTER; GROSSBERG, 1998). O teste de vigilância é realizado confrontando-se o padrão de entrada com os pesos *top-down* correspondentes ao neurônio vencedor. Caso passe no teste (ocorre a ressonância), o neurônio vencedor assimila as informações do padrão de entrada, caso contrário a busca se repete até que todos os neurônios sejam avaliados. Se nenhum candidato passar no teste, um novo neurônio é criado (adaptação) que então aprende o padrão de entrada. Este mecanismo de busca é a base do princípio de estabilidade das redes ART (CARPENTER; GROSSBERG, 1998).

O critério de similaridade é controlado pelo parâmetro de vigilância ρ. Uma baixa vigilância permite uma maior generalização com categorias mais abrangentes, memória mais abstrata e consequentemente demanda menos neurônios na camada F_2. Por outro lado, uma alta vigilância promove uma generalização limitada, categorias refinadas e uma memória detalhada. Na condição de extrema vigilância, cada categoria irá representar um único padrão de entrada (CARPENTER; GROSSBERG, 1998).

A rede pode operar, combinadamente, no modo de aprendizado rápido ou lento, controlado pelo parâmetro de taxa de aprendizagem $\beta \in (0, 1]$. Na ocorrência de eventos raros, no qual o mecanismo de busca determina a criação de uma nova categoria, a rede opera no modo de aprendizado rápido (valores de β próximos a 1). Com isso o sistema assimila a nova informação rapidamente, sem comprometer nenhuma categoria existente, e consequentemente tem condições de generalizar frente a padrões semelhantes. No modo de aprendizado lento (valores intermediários de β), a rede previne a deleção de informações importantes frente a padrões com ruídos ou com informações incompletas. Desta forma apenas uma mudança persistente nos padrões de entrada podem promover, a longo prazo, alterações significativas nas categorias preexistentes da rede (DAVID; RAJASEKARAN, 2009).

3.2 *Fuzzy* ART

A teoria ART evoluiu como uma série de modelos matemáticos de redes neurais que podem operar com treinamento supervisionado e não supervisionado. O primeiro deles, denominado ART 1 (CARPENTER; GROSSBERG, 1987a), era capaz de reco-

nhecer categorias de uma sequência arbitrária de padrões de entradas binárias. ART 2 (CARPENTER; GROSSBERG, 1987b), por sua vez, era a versão analógica de ART 1, capaz categorizar padrões com valores reais. Inserindo melhorias na ART 2, Carpenter, Grossberg e Rosen (1991a) propõem a ART 2-B que apresenta um aumento considerável na velocidade de processamento se comparada à versão original. Em Carpenter e Grossberg (1990) é introduzido o modelo ART 3, onde vários módulos de ART 2 são dispostos de forma hierárquica. Nesta configuração, os sinais de saída camada F_2 são entradas para a camada F_1 do módulo superior. O mecanismo de busca sofreu uma reformulação, para atender à nova arquitetura, baseada nas propriedades químicas dos neurotransmissores fisiológicos.

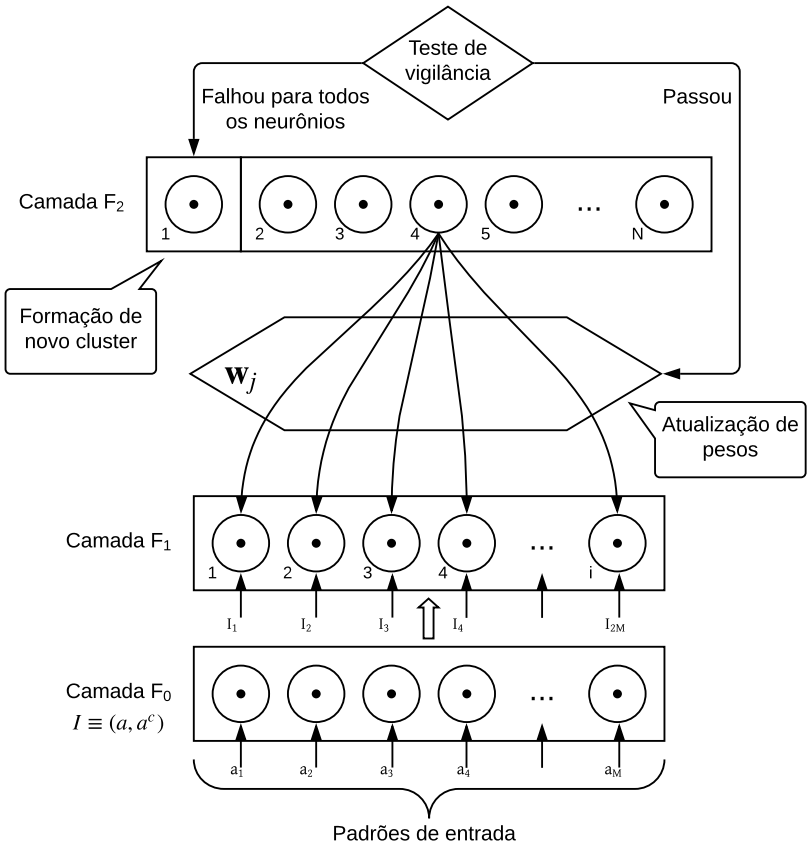
A primeira rede neural de treinamento supervisionado, derivada da teoria ART, foi a ARTMAP (CARPENTER; GROSSBERG; REYNOLDS, 1991). É composta por dois módulos ART 1, ARTa e ARTb, além do módulo *map field* que associa as categorias de ARTa com as categorias de ARTb. No entanto, como esta rede é baseada na rede ART 1, ela opera somente com padrões binários. Para superar esta limitação Carpenter et al. (1992) introduziram a FAM, de arquitetura idêntica à ARTMAP, com exceção aos módulos ARTa e ARTb que foram substituídos pela rede *Fuzzy ART* (FA) capaz de categorizar padrões de entrada com valores reais.

Dado que FAM foi uma das redes neurais analisadas no experimento desta pesquisa e que por sua vez é composta por dois módulos FA, esta seção limita-se em detalhar apenas o funcionamento da rede FA.

3.2.1 Arquitetura

FA é uma rede neural capaz de categorizar uma sequência arbitrária de padrões analógicos ou binários. Sua arquitetura se assemelha à arquitetura simplificada apresentada na seção 3.1. Porém, na FA os pesos *bottom-up* e *top-down* são substituídos por apenas um vetor de pesos $\mathbf{W}_j$ (Figura 17) que representa o protótipo da categoria do j-ésimo neurônio da camada F_2. Outra diferença é a inclusão da camada F_0 na arquitetura, responsável pelo pré processamento do padrão de entrada.

Figura 17 – Arquitetura da rede neural Fuzzy ART com codificação de complemento na camada F_0



Fonte – Adaptado de Rajasekaran e Pai (2017)

3.2.2 Algoritmo de treinamento

A rede FA se diferencia da ART 1 pela inclusão do operador AND ($\wedge$), da teoria de conjuntos *fuzzy*, em substituição ao operador de interseção ($\cap$). Esta melhoria possibilitou à rede FA a categorização de padrões analógicos. Nesta seção é apresentado o seu algoritmo de treinamento.

1. **Inicialização dos pesos:** O protótipo que define cada categoria $j \in (1, \ldots, N)$ é um vetor de pesos $\mathbf{w}_j \equiv (w_{j1}, \ldots, w_{j2M})$, onde N é o número de categorias e $2M$ é o tamanho do vetor de entrada após a codificação de complemento. Os pesos são inicializados $w_{j1} = \cdots = w_{j2M} = 1$.

2. **Leitura dos hiperparâmetros:** Parâmetro de escolha $\alpha > 0$; taxa de aprendizagem $\beta \in (0, 1]$ e parâmetro de vigilância $\rho \in (0, 1]$.

3. **Leitura do vetor de entrada:** Cada padrão de entrada **a** é um vetor M dimensional $(a_1, \ldots, a_M)$ onde cada componente a_i assume um valor do intervalo $[0, 1]$.

4. **Normalização do vetor de entrada:** Com o objetivo de se evitar a proliferação de categoria (MOORE, 1989), o vetor de entrada **a** é normalizado utilizando a codificação de complemento:

$$\mathbf{I} \equiv (\mathbf{a}, \mathbf{a}^c) \equiv (a_1, \ldots, a_M, a_1^c, \ldots, a_M^c) \tag{3.1}$$

onde $a_i^c = 1 - a_i$. Logo o vetor **I** será $2M$ dimensional.

5. **Escolha da categoria:** Para cada categoria j, a função de escolha T_j é definida como:

$$T_j = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|} \tag{3.2}$$

onde o operador *fuzzy* $\wedge$ é definido por:

$$(\mathbf{x} \wedge \mathbf{y})_i \equiv min(x_i, y_i) \tag{3.3}$$

e a norma $|\cdot|$ é definida por:

$$|\mathbf{x}| \equiv \sum_{i=1}^{M} |x_i| \tag{3.4}$$

A categoria escolhida representada pelo índice J é dada por:

$$T_J = max\{T_j : j = 1, \ldots, N\} \tag{3.5}$$

6. **Ressonância ou *reset*:** A ressonância ocorre quando a categoria escolhida torna verdadeira a desigualdade:

$$\frac{|\mathbf{I} \wedge \mathbf{w}_J|}{|\mathbf{I}|} \geq \rho \tag{3.6}$$

caso contrário ocorre o *reset*: o valor da função de escolha T_J é igualada a -1 e uma nova categoria J é escolhida por (3.5). O processo se repete até que ocorra a ressonância. Caso nenhuma categoria atenda ao critério de vigilância, uma nova é criada e utilizada para aprender o padrão de entrada.

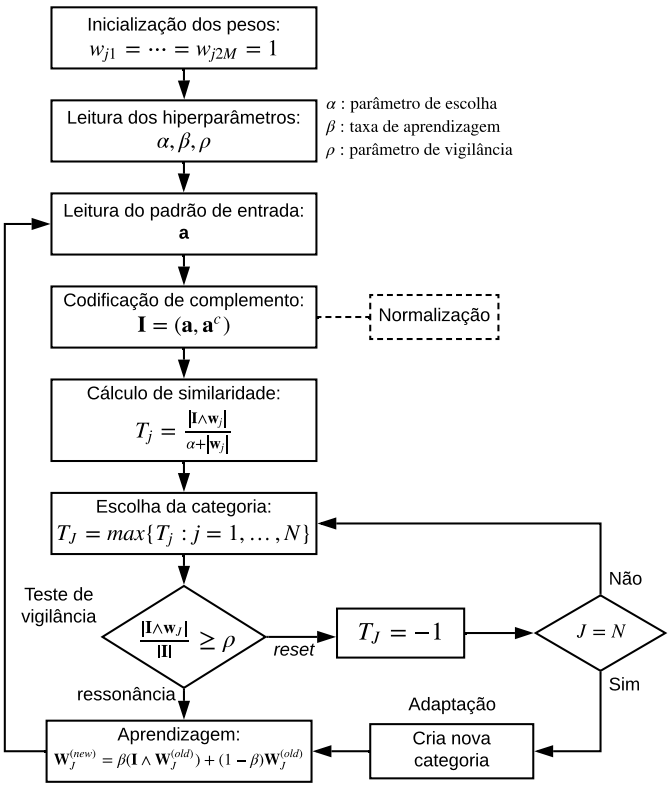
7. **Aprendizagem :** O vetor $\mathbf{W}_j$ é atualizado de acordo com a equação:

$$\mathbf{W}_J^{(new)} = \beta(\mathbf{I} \wedge \mathbf{W}_J^{(old)}) + (1 - \beta)\mathbf{W}_J^{(old)} \tag{3.7}$$

8. **Aprendizado rápido e recodificação lenta:** Como detalhado na seção 3.1 a rede pode operar combinando o aprendizado rápido e a recodificação lenta. Ao criar uma categoria, a rede utiliza o aprendizado rápido ($\beta = 1$) para assimilar a nova informação. Para atualização de categorias existentes a rede chaveia para o modo de recodificação lenta (valores intermediários de β) prevenindo que o aprendizado já assimilado seja corrompido.

A Figura 18 apresenta o fluxograma de treinamento da rede neural FA, com todos os passos do algoritmo descrito anteriormente.

Figura 18 – Fluxograma de treinamento da rede neural Fuzzy ART



Fonte – Adaptado de Moreno (2010), Serrano-Gotarredona, Linares-Barranco e Andreou (1998)

3.3 *Euclidean* ART

Euclidean ART (EA) é uma rede neural de treinamento não supervisionado, utilizada para classificação de padrões de entradas analógicas. Ela se diferencia de FA pela inclusão da distância Euclidiana como medida de similaridade ao invés do operador AND *fuzzy*. Segundo os autores Kenaya e Cheok (2008a), EA foi elaborada para superar duas deficiências da FA: sensibilidade a ruídos e representações ineficientes das categorias *fuzzy*.

Não havendo alterações na topologia de rede em relação à FA, nesta seção será apresentado somente o algoritmo de treinamento da EA.

3.3.1 Algoritmo de treinamento

1. **Leitura do hiperparâmetro R_{th}:** Similar ao parâmetro ρ da rede FA, o parâmetro R_{th} determina o raio da hiperesfera que define o tamanho da categoria. Valores baixos de R_{th} implicam em uma quantidade maior de categorias e um menor número de membros em cada uma delas.

2. **Leitura e normalização do padrão de entrada:** Os componentes dos padrões de entrada são reescalados para o intervalo $[0, 1]$ utilizando a normalização min-max (ver seção 2.2.4.2.1). Posteriormente o padrão é normalizado utilizando a codificação de complemento, da mesma forma que ocorre na rede FA.

3. **Escolha da categoria:** Para cada categoria j é calculada a distância euclideana em relação ao padrão de entrada $\mathbf{I}$:

$$d(\mathbf{I}, \mathbf{W}_j) = \sqrt{(\mathbf{I} - \mathbf{W}_j)^T(\mathbf{I} - \mathbf{W}_J)} \tag{3.8}$$

A categoria J é escolhida tal que:

$$d(\mathbf{I}, \mathbf{W}_J) = min[d(\mathbf{I}, \mathbf{W}_j)], \quad j = 1, \ldots, N \tag{3.9}$$

4. **Ressonância:** A ressonância ocorre se $d(\mathbf{I}, \mathbf{W}_J) \leq R_{th}$. Neste caso o padrão $\mathbf{I}$ é adicionado à categoria J. Caso contrário uma nova categoria é criada cujo centro da hiperesfera é o padrão $\mathbf{I}$.

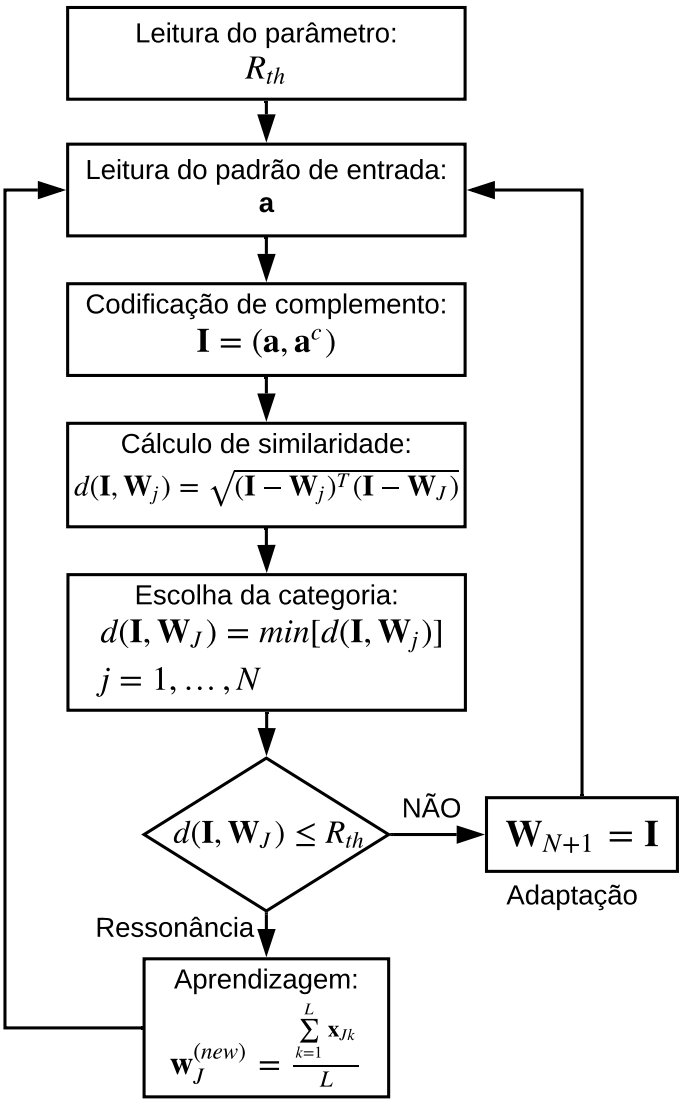
5. **Aprendizagem:** A adaptação é efetuada recalculando-se o centro da hiperesfera através da média aritmética de todos os padrões pertencentes à categoria:

$$\mathbf{w}_J^{(new)} = \frac{\sum_{k=1}^{L} \mathbf{x}_{Jk}}{L} \tag{3.10}$$

onde $\mathbf{x}_{Jk}$ é o k-ésimo membro da categoria J e L é a quantidade de membros da categoria J.

O fluxograma do procedimento de treinamento descrito acima está representado na Figura 19.

Figura 19 – Fluxograma de treinamento da rede neural Euclidean ART



Fonte – Adaptado de Kenaya e Cheok (2008a)

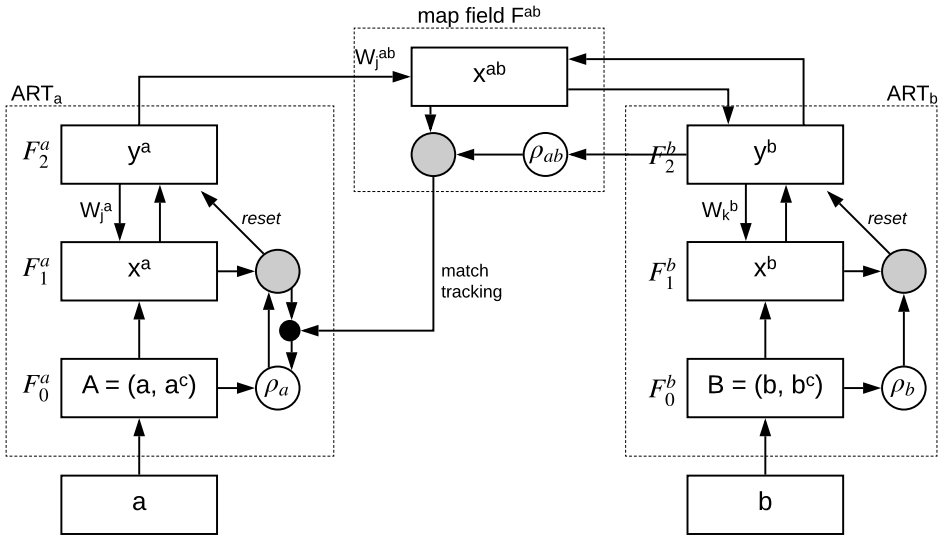
3.4 *Fuzzy* ARTMAP

FAM (CARPENTER et al., 1992) é uma rede neural de treinamento supervisionado e incremental capaz de aprender o mapeamento entre pares de padrões $(\mathbf{a}, \mathbf{b})$, onde $\mathbf{a}$ é um vetor de entrada e $\mathbf{b}$ um vetor que representa a saída desejada.

3.4.1 Arquitetura

A arquitetura da rede FAM é composta por dois módulos FA, ART_a e ART_b, um módulo *map field* e um mecanismo de *match tracking* (Figura 20).

Figura 20 – Arquitetura da rede neural *Fuzzy* ARTMAP



Fonte – Carpenter et al. (1992)

Durante o treinamento o módulo ART_a recebe o vetor **a**, e o módulo ART_b recebe o vetor de saída desejada **b**. Os dois módulos são interligador pelo módulo inter-ART F^{ab} chamado de *map field* que associa a categoria j da camada F_2^a do módulo ART_a à categoria k da camada F_2^b do módulo ART_b. O padrão **b** é categorizado pelo módulo ART_b, representado pela categoria K, conforme o procedimento de aprendizado abordado na seção 3.2.2. O aprendizado do padrão **a** pelo módulo ART_a é regulado pelo mecanismo de *match tracking*: se a categoria J escolhida no módulo ART_a for uma categoria nova, a associação entre J e K é memorizada no *map field* atribuindo-se $w_{JK}^{ab} = 1$. Se J for uma categoria existente e houver o relacionamento $w_{JK}^{ab} = 1$ no *map field* então a categoria J aprende o padrão **a**. Por outro lado, se J for uma categoria existente, mas não ocorre o relacionamento entre J e K no *map field*, o parâmetro de vigilância ρ_a do módulo ART_a é incrementado e a busca por uma nova categoria J é iniciada.

3.4.2 Algoritmo de treinamento

A seguir é descrito o algoritmo do processo de treinamento da rede FAM:

1. **Leitura dos hiperparâmetros:**

- $\bar{\rho}_a$: parâmetro de vigilância base do módulo ART_a
- ρ_b : parâmetro de vigilância do módulo ART_b

- $\alpha > 0$: parâmetro de escolha
- $\beta \in (0, 1]$: taxa de aprendizagem

2. **Leitura dos padrões de entrada:** Leitura dos padrões de entrada $(\mathbf{a}, \mathbf{b})$ onde $a_i \in [0, 1]$ e $b_i \in [0, 1]$.

3. **Codificação de complemento:** Os padrões $(\mathbf{a}, \mathbf{b})$ são normalizados utilizando a codificação de complemento:

$$\begin{aligned} \mathbf{A} &= (\mathbf{a}, \mathbf{a}^c) \quad | \quad a_i^c = 1 - a_i \\ \mathbf{B} &= (\mathbf{b}, \mathbf{b}^c) \quad | \quad b_i^c = 1 - b_i \end{aligned} \tag{3.11}$$

4. **Atribuição $\rho_a = \bar{\rho}_a$:** a variável $\bar{\rho}_a$ lida no início do algoritmo representa o parâmetro de vigilância mínimo do módulo ART_a. Este passo reinicia a variável ρ_a à cada apresentação um novo par de padrões $(\mathbf{a}, \mathbf{b})$, uma vez que a mesma é incrementada de forma autônoma pelo mecanismo de *match tracking*.

5. **Categorização do padrão B:** o padrão $\mathbf{B}$ é classificado no módulo ART_b pela categoria K, conforme procedimento descrito na seção 3.2.2.

6. **Escolha da categoria no módulo ART_a:** a categoria J no módulo ART_a é escolhida para classificar o padrão $\mathbf{A}$, conforme procedimentos descritos nos passos 5 e 6 do algoritmo de aprendizagem da rede FA apresentado na seção 3.2.2.

7. **Mecanismo *match tracking*:** se J é uma nova categoria, então o padrão A é assimilado no modo de aprendizado rápido ($\beta = 1$) conforme equação apresentada no passo 7 do algoritmo de aprendizado da rede FA. Vá para o passo 8 a seguir. Se J é uma categoria existente, outras duas situações podem ocorrer:

- se $w_{JK}^{ab} \neq 1$ (ou seja, não ocorre o mapeamento entre J e K no módulo inter-ART): o parâmetro ρ_a é incrementado de acordo com (3.12) e uma nova busca pela categoria J é iniciada conforme o passo 6.

$$\rho_a = \frac{|A \wedge w_J^a|}{|A|} + \epsilon \tag{3.12}$$

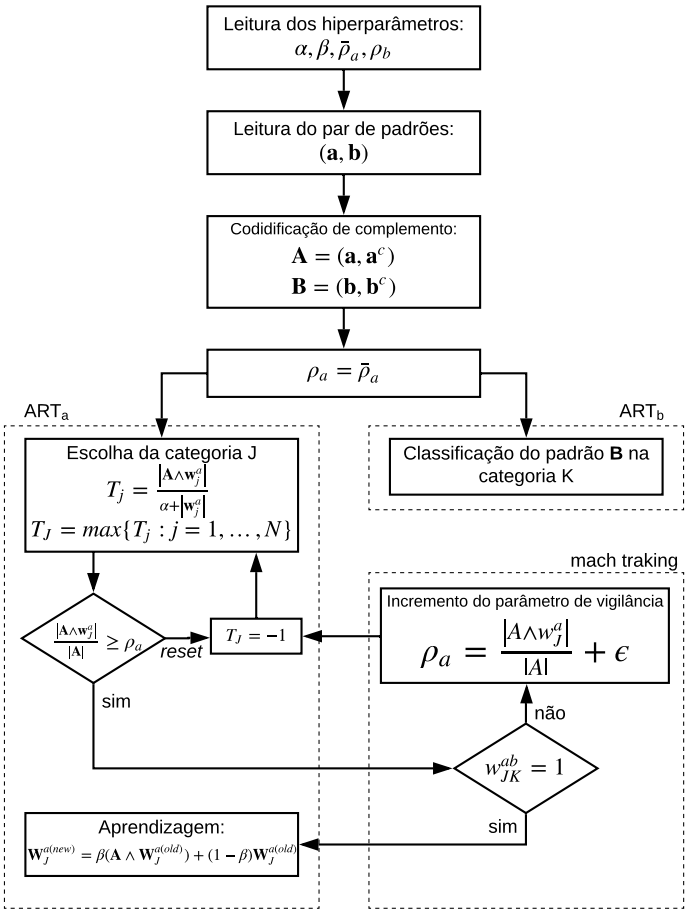
  onde ϵ é uma constante suficientemente pequena para que ρ_a cresça de forma gradativa;

- se $w_{JK}^{ab} = 1$: a categoria J do módulo ART_a aprende o padrão A, no modo de recodificação lenta, conforme passo 7 do algoritmo de aprendizado da rede FA. Volte para o passo 2 se houver mais padrões de treinamento.

8. **Aprendizagem no módulo** *inter-ART*: a relação entre a categoria J do módulo ART_a e a categoria K do módulo ART_b é memorizada atribuindo-se $w_{JK}^{ab} = 1$. Volte para o passo 2 se houver mais padrões de treinamento.

O fluxograma do algoritmo de treinamento apresentado acima, é exibido na Figura 21.

Figura 21 – Fluxograma do algoritmo de treinamento da rede neural *Fuzzy* ARTMAP



Fonte – Adaptado de Serrano-Gotarredona, Linares-Barranco e Andreou (1998)

3.5 *Euclidean* ARTMAP

EAM (KENAYA; CHEOK, 2008b) é uma rede neural de treinamento supervisionado e incremental. De arquitetura similar à FAM, se diferencia da mesma pela substituição dos módulos FA pelo módulos EA. A seguir é apresentado o algoritmo de treinamento desta rede, que se caracteriza principalmente pela utilização da distância euclidiana para o cálculo de similaridade.

3.5.1 Algoritmo de treinamento

1. **Leitura dos hiperparâmetros:**

   - $\bar{R}_{tha}$: tamanho base do raio das hiperesferas das categorias formadas no módulo ART_a
   - R_{thb} : tamanho do raio das hiperesferas das categorias formadas no módulo ART_b

2. **Leitura dos padrões de entrada:** Leitura dos padrões de entrada $(\mathbf{a}, \mathbf{b})$ onde $a_i \in [0, 1]$ e $b_i \in [0, 1]$.

3. **Codificação de complemento:** Os padrões $(\mathbf{a}, \mathbf{b})$ são normalizados utilizando a codificação de complemento:

$$\begin{aligned} \mathbf{A} &= (\mathbf{a}, \mathbf{a}^c) \quad | \quad a_i^c = 1 - a_i \\ \mathbf{B} &= (\mathbf{b}, \mathbf{b}^c) \quad | \quad b_i^c = 1 - b_i \end{aligned} \tag{3.13}$$

4. **Atribuição $R_{tha} = \bar{R}_{tha}$:** a variável $\bar{R}_{tha}$ lida no início do algoritmo representa o tamanho máximo do raio das hiperesferas das categorias do módulo ART_a. Este passo reinicia a variável R_{tha} à cada apresentação um novo par de padrões $(\mathbf{a}, \mathbf{b})$, uma vez que a mesma é decrementada de forma autônoma pelo mecanismo de *match tracking*.

5. **Cálculo das distâncias em ART_a:** A distância euclidiana do padrão $\mathbf{A}$ em relação ao centro de cada hiperesfera de cada categoria é calculada segundo (3.14).

$$da(\mathbf{A}, \mathbf{W}_j^a) = \sqrt{(\mathbf{A} - \mathbf{W}_j^a)^T (\mathbf{A} - \mathbf{W}_J^a)} \tag{3.14}$$

6. **Escolha das categorias:** Encontre $da(\mathbf{A}, \mathbf{W}_J^a)$ e $db(\mathbf{B}, \mathbf{W}_K^b)$, tais que:

$$da(\mathbf{A}, \mathbf{W}_J^a) = min(da(\mathbf{A}, \mathbf{W}_j^a)), \quad j = 1, \ldots, N \tag{3.15}$$

$$da(\mathbf{B}, \mathbf{W}_K^b) = \sqrt{(\mathbf{B} - \mathbf{W}_K^a)^T (\mathbf{B} - \mathbf{W}_K^a)} \tag{3.16}$$

onde N é o número de categorias de ART_a e K é uma categoria em ART_b que foi mapeada à categoria J no módulo inter-ART (pesos w^{ab}).

7. **Mecanismo *match tracking*:**

se $da(\boldsymbol{A}, \boldsymbol{W}_J^a) \leq R_{tha}$ **então**

- **se** $db(\boldsymbol{B}, \boldsymbol{W}_K^b) \leq R_{thb}$ **então**
  - **aprendizagem dos padrões:**
  - incluir o padrão **A** na categoria J do módulo ART_a;
  - incluir o padrão **B** na categoria K do módulo ART_b;
  - atualização das categorias J e K conforme passo 5 do algoritmo de aprendizagem da rede EA;
- **senão**
  - **se** $R_{tha} > 0$ **então**
    - $R_{tha} = R_{tha} - \epsilon \quad | \quad 0 < \epsilon << 1$;
    - vá para o passo 5;
  - **fim**
- **fim**

senão

- formação de novas categorias $N+1$ e $P+1$ em ART_a e ART_b respectivamente;
- $\mathbf{W}_{N+1}^a = A$; (o centro da hiperesfera da categoria N+1 é o padrão A)
- $\mathbf{W}_{P+1}^b = B$; (o centro da hiperesfera da categoria P+1 é o padrão B)
- $\mathbf{W}_{N+1}^{ab} = P+1$; (mapeamento da categoria $N+1$ na categoria $P+1$)

fim

volte para o passo 2 se existirem mais padrões de treinamento;

4 SÉRIES TEMPORAIS

Uma série temporal é um conjunto ordenado de medições de um determinado fenômeno (realizações), aferidos em intervalos de tempos regulares (de hora em hora, semanal, mensal, anualmente, etc). Para exemplificar o conceito, podemos listar as seguintes séries temporais:

a) aferições diárias da qualidade do ar de uma região;

b) quantidade de acessos mensais a um determinado *website*;

c) consumo mensal de energia elétrica de uma universidade;

d) índice pluviométrico aferido mensalmente em uma cidade;

e) medidas de uma variável de controle de um processo.

Levando-se em conta a maneira com que os dados são registrados, elas podem ser classificadas como contínuas, quando as observações são aferidas continuamente em um intervalo de tempo ou discretas, quando aferidas em intervalos de tempos regulares. Séries temporais contínuas podem ser discretizadas realizando amostragens em intervalos de tempos iguais. Em outros casos a série contínua pode ser discretizada agregando-se os dados em intervalos de tempos igualmente espaçados. Esta classificação é independente da natureza da variável observada. Uma série temporal pode ser classificada como contínua mesmo quando a variável observada for discreta, ou ainda classificada como discreta mesmo quando a variável observada for contínua (CHATFIELD, 2003).

A série temporal também é classificada como univariada, quando as observações são realizadas tomando-se apenas uma variável, ou multivariada quando a série temporal principal é acompanhada de outras séries (variáveis explicativas) que contribuem para o entendimento do comportamento da série principal (CHATFIELD, 2003).

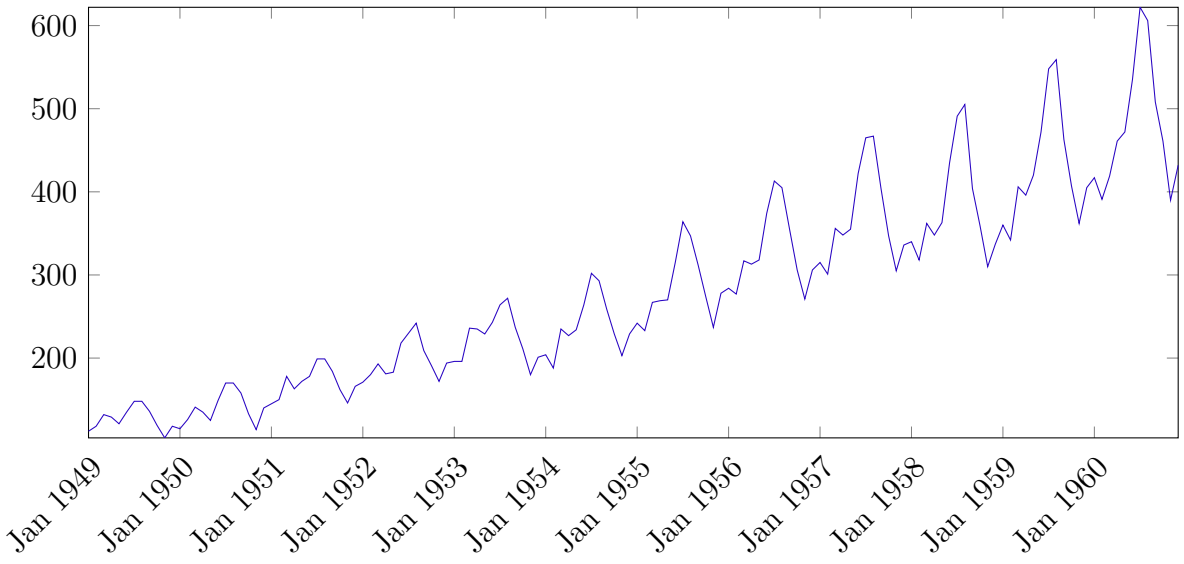
A principal característica que permite que uma série temporal seja analisada, é que as observações sucessivas possuem dependência entre si. Segundo Bisgaard e Kulahci (2011) a esta característica da-se o nome de autocorrelação. Devido a esta propriedade, valores futuros podem ser previstos levando-se em consideração os valores históricos. A série temporal é dita determinística quando os valores futuros poderem ser previstos de forma exata. No entanto, a maioria das séries temporais presentes em situações reais possuem um elemento estocástico. Nestes casos a previsão exata dos valores futuros se

torna impossível e assumem uma distribuição de probabilidades calculada tomando-se os valores realizados no passado (CHATFIELD, 2003).

Analisar uma série temporal, para Adhikari e Agrawal (2013), compreende métodos e procedimentos que melhoram o entendimento sobre a série com o objetivo de encontrar um modelo que a represente com o máximo de precisão possível.

Segundo Chatfield (2003), a análise das séries temporais tem por objetivos: descrição, explicação, previsão e controle. Ao iniciar o estudo de uma série temporal o primeiro passo é desenhar o gráfico dos valores observados ao longo do tempo. O exame visual do gráfico pode fornecer instantaneamente algumas informações relevantes sobre o comportamento da série temporal, tornando-se uma ferramenta descritiva importante. A título de exemplo, a série temporal representada no Gráfico 1 mostra claramente uma sazonalidade anual com picos de passageiros nos meses de julho e agosto. O gráfico também mostra uma tendência de crescimento ao longo dos anos. Além de mostrar a sazonalidade e a tendência, o gráfico da série temporal pode indicar também a presença de *outliers* que são observações desproporcionais se comparadas com as demais.

Gráfico 1 – Passageiros de linhas aéreas internacionais: totais mensais (em milhares) de Janeiro de 1949 - Dezembro 1960.



Fonte – (BOX et al., 2015)

Em outras situações o objetivo é apenas explicar o comportamento de uma série temporal em função de outra(s), por exemplo, estudar o efeito que uma variável macroeconômica teria no valor das ações de uma determinada empresa. Outro importante objetivo

ao se analisar uma série temporal é a previsão de valores futuros. Neste caso se define um horizonte h de previsão, e então o modelo matemático da série é empregado para gerar os possíveis valores futuros nos instantes $t+1, t+2, \ldots, t+h$, onde t é o instante da última observação da série.

A análise de série temporal também pode ser aplicada quando o objetivo é melhorar o controle sobre processos físicos ou econômicos. Ao gerar uma série das medições da qualidade de um processo, o controle sobre o mesmo poderia atuar de forma com que ele se mantivesse em um nível alto de desempenho. A predição da série da qualidade do processo poderia apontar para uma possível degradação, permitindo que ações corretivas possam ser tomadas.

4.1 Componentes da série temporal

As séries temporais, de modo geral, apresentam basicamente os seguintes componentes que afetam o seu comportamento: tendência, ciclos, sazonalidade e irregularidades. A tendência mostra como a série se comporta a longo prazo. Séries como crescimento populacional e número de residência em uma cidade apresentam tendência ascendente. Por outro lado, séries como taxas de mortalidade e doenças infecciosas possuem tendência descendente.

O componente sazonal representa as variações que ocorrem em períodos regulares. O Gráfico 1 mostra um aumento importante das vendas de passagens aéreas nos meses de julho e agosto a cada ano. Outro exemplo seria o aumento de consumo de energia elétrica no inverno.

O componente cíclico difere do componente sazonal por não possuir um período regular para ocorrer, por exemplo, variações econômicas em períodos de crescimento e recessão. A duração de cada ciclo pode se estender por períodos longos, normalmente por dois ou mais anos.

As variações irregulares se caracterizam por não possuírem um padrão regular para acontecerem. São causadas por incidentes como guerras, greves e eventos climáticos como terremotos, furacões, etc.

Dado os componentes da série temporal, basicamente dois tipos de modelos são

utilizados para representá-las: os modelos aditivos (4.1) e multiplicativos (4.2).

$$Y(t) = T(t) + S(t) + C(t) + I(t) \tag{4.1}$$

$$Y(t) = T(t) \times S(t) \times C(t) \times I(t) \tag{4.2}$$

onde $Y(t)$ representa o valor da série no instante t e $T(t), S(t), C(t)$ e $I(t)$ são respectivamente os componentes tendência, sazonalidade, cíclico e irregularidade. No modelo aditivo assume-se que os quatro componentes são independentes entre si, ao contrário do modelo multiplicativo que entende que os componentes podem ser dependentes um do outro (ADHIKARI; AGRAWAL, 2013).

4.2 Variável aleatória

Seja o espaço amostral Ω definido como o conjunto de todos os possíveis resultados que um evento pode assumir. Uma variável aleatória x (4.3) é uma função que mapeia cada elemento $\omega \in \Omega$ a um valor em $\mathbb{R}$ (ALBUQUERQUE; FORTES; FINAMORE, 2008).

$$\begin{aligned} x: \quad & \Omega \longmapsto \mathbb{R} \\ & \omega \longmapsto x(\omega) \end{aligned} \tag{4.3}$$

Para exemplificar o conceito, considere o experimento composto por três lançamentos consecutivos de uma moeda, onde K é o resultado para cara e C para coroa. Neste caso o espaço amostral é dado por (4.4).

$$\begin{aligned} \Omega = \{&(C,C,C),(C,C,K),(C,K,C),(C,K,K), \\ &(K,C,C),(K,C,K),(K,K,C),(K,K,K)\} \end{aligned} \tag{4.4}$$

Seja x uma variável aleatória que assume o valor 1 quando os lançamentos possuírem duas caras e 0 caso contrário. Portanto, os possíveis valores para x estão listados em (4.5).

$$\begin{aligned} x((C,C,C)) &= 0 & x((K,C,C)) &= 0 \\ x((C,C,K)) &= 0 & x((K,C,K)) &= 1 \\ x((C,K,C)) &= 0 & x((K,K,C)) &= 1 \\ x((C,K,K)) &= 1 & x((K,K,K)) &= 1 \end{aligned} \tag{4.5}$$

4.2.1 Função distribuição de probabilidade de uma variável aleatória

Denomina-se evento, o conjunto A_X definido por (4.6).

$$A_X = \{\omega \in \Omega, X \in \mathbb{R} : x(\omega) \leq X\} \tag{4.6}$$

A função distribuição de probabilidade (FDP) de uma variável aleatória x é dada por (4.7).

$$\begin{aligned} F_x : \quad & \mathbb{R} \longmapsto \mathbb{R} \\ & X \longmapsto F_x(X) = P(A_X) \end{aligned} \tag{4.7}$$

Ou seja, $F_x(X)$ é a probabilidade que a variável aleatória x seja menor ou igual a X (ALBUQUERQUE; FORTES; FINAMORE, 2008).

O conceito pode ser estendido quando se considera um conjunto de variáveis aleatórias. A função distribuição de probabilidade conjunta das variáveis aleatórias $x_1, \ldots, x_n$ é definida em (4.8).

$$\begin{aligned} F_{x_1,\ldots,x_n} : \quad & \mathbb{R}^n \longmapsto \mathbb{R} \\ & X_1, \ldots, X_n \longmapsto F_{x_1,\ldots,x_n}(X_1, \ldots, X_n) = P(A_{X_1}, \ldots, A_{X_n}) \end{aligned} \tag{4.8}$$

onde $A_{X_1}, \ldots, A_{X_n}$ são eventos definidos em (4.6). Do que foi apresentado podemos concluir que a função distribuição de probabilidade conjunta das variáveis aleatórias $x_1, \ldots, x_n$ é a probabilidade de que sejam menores ou iguais, respectivamente, a $X_1, \ldots, X_n$ ao mesmo tempo.

4.3 Séries temporais e processo estocástico

Dado um conjunto qualquer T (na maioria dos casos T está relacionado ao tempo), um processo estocástico é o conjunto $Z = \{Z(t), t \in T\}$ tal que $Z(t)$ é uma variável aleatória (MORETTIN; de Castro Toloi, 2006). Representa-se o processo por $Z(t)$ se t é contínuo, isto é assume valores de um determinado intervalo contínuo, por exemplo: $T = \{t : 0 \leq t < \infty\}$ e por Z_t caso t é discreto (geralmente $t = 0, \pm 1, \pm 2, \ldots$) (CHATFIELD, 2003).

Para cada instante $t \in T$, a variável aleatória $Z(t)$ pode assumir qualquer valor de seu contradomínio. No entanto, se fixarmos $w \in \Omega$ em cada instante t, temos uma realização ou trajetória do processo estocástico. Ao conjunto de todas as possíveis realizações do processo dá-se o nome de *ensemble* (MORETTIN; de Castro Toloi, 2006). Sendo assim, uma série temporal pode ser definida como uma realização em particular do processo estocástico (CHATFIELD, 2003).

Um processo estocástico é completamente descrito pela distribuição de probabilidade conjunta das variáveis aleatórias $Z(t_1), Z(t_2), \ldots, Z(t_k)$ para qualquer conjunto de instantes

$t1, t2, \ldots, tk$ e qualquer k. Entretanto, para problemas práticos esta abordagem não é adotada devido a sua complexidade elevada. Uma maneira mais simples de descrever um processo estocástico é considerar momentos do processo. Os momentos comumente usados são os de primeira e segunda ordem, chamados respectivamente de função média (4.9) e função autocovariância (4.11) (CHATFIELD, 2003).

$$\mu_{x_t} = E(x_t) = \int_{-\infty}^{\infty} x f_t(x) dx, \tag{4.9}$$

$$f_t(x) = \frac{\partial F_t(x)}{\partial x} \tag{4.10}$$

A função média é o valor esperado para a variável aleatória x_t (SHUMWAY; STOFFER, 2017). (4.10) é função densidade de probabilidade obtida através da derivação da função distribuição de probabilidade (4.7). A mesma representa a probabilidade de a variável aleatória assumir o valor x.

$$\gamma_x(s,t) = cov(x_s, x_t) = E[(x_s - \mu_s)(x_t - \mu_t)] \tag{4.11}$$

A função de autocovariância (4.11) mede a dependência linear entre dois pontos da série, observados em momentos diferentes. A variância (4.12) é um caso especial da autocovariância quando $s = t$.

$$\gamma_x(t,t) = E[(x_t - \mu_t)^2] = var(x_t) \tag{4.12}$$

No entanto, a magnitude da função de autocovariância depende da unidade em que a variável aleatória é medida. Portanto, para fins de comparação, padroniza-se a função de autocovariância produzindo-se a função de autocorrelação, definida por (4.13).

$$\rho(s,t) = \frac{\gamma(s,t)}{\sqrt{\gamma(s,s)\gamma(t,t)}} \tag{4.13}$$

4.4 Conceito de estacionariedade

Um processo estocástico é dito estacionário quando suas propriedades estatísticas não se alteram como tempo. Muitos métodos presumem esta condição para a construção de modelos de previsão (ADHIKARI; AGRAWAL, 2013).

O processo $Z = \{Z(t), t = 0, 1, 2, \ldots\}$ é estritamente estacionário quando a distribuição conjunta (4.8) de $\{Z(t1), Z(t2), \ldots, Z(tk)\}$ é a mesma de $\{Z(t1+T), Z(t2+T), \ldots, Z(tk+T)\}$ para todo $t1, t2, \ldots, tk \in t$ e T. Ou seja a distribuição conjunta não é

afetada pelo deslocamento T de tempo e depende somente dos instantes $t1, t2, \ldots, tk$. No entanto, na prática, se define a estacionariedade de um processo de uma maneira menos restrita considerando-se apenas os primeiros dois momentos da série. Neste caso o processo é classificado como fracamente estacionário e deve atender às seguintes propriedades:

a) a função média μ_{x_t} definida em (4.9), é constante para todo t;

b) a função de autocovariância $\gamma_x(s,t)$ definida em (4.11), depende apenas da diferença $|s-t|$ (SHUMWAY; STOFFER, 2017).

5 MATERIAL E MÉTODOS

Este trabalho comparou o desempenho das RNA FAM (CARPENTER et al., 1992) e EAM (KENAYA; CHEOK, 2008b) na tarefa de previsão de séries temporais com treinamento continuado. Como linha de base foram utilizados os algoritmos KNN (RASCHKA; MIRJALILI, 2017) e SNAIVE (HYNDMAN; ATHANASOPOULOS, 2018). Modelos de linha de base são de simples implementação e fáceis de serem interpretados. Além disso, fornecem uma referência de performance para comparação com algoritmos mais sofisticados.

Para o treinamento e avaliação dos modelos de previsão, foi utilizada a série temporal histórica formada pela demanda diária de refeições servidas pelo RU da UFU. O RU é constituído por quatro unidades distribuídas nos campi da universidade conforme lista do Quadro 3. Cada unidade oferece à comunidade acadêmica três modalidades de refeições: café da manhã, almoço e jantar, com exceção à unidade Glória que serve apenas almoço.

Quadro 3 – Unidades do restaurante universitário da Universidade Federal de Uberlândia

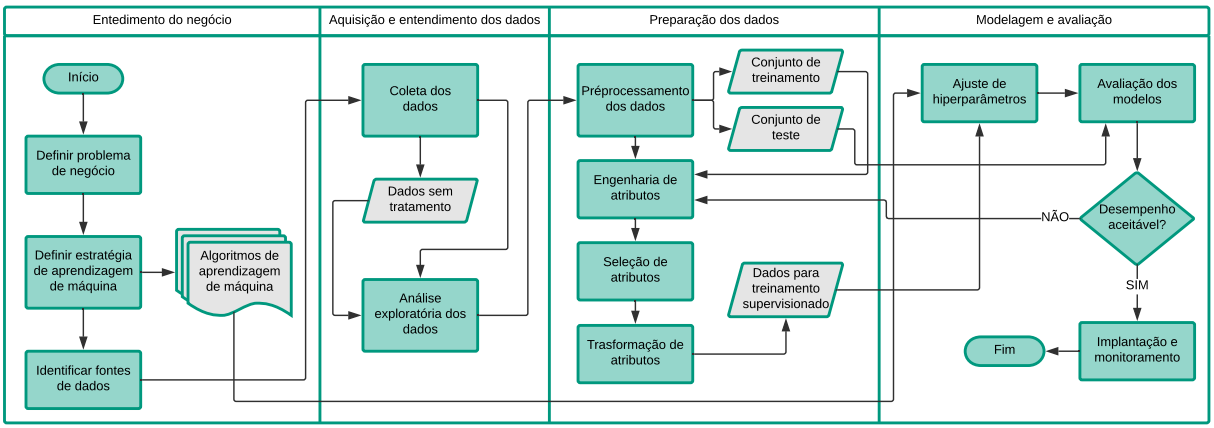
Nome da unidade	**Localização**
RU Umuarama	Campus Umuarama, Uberlândia, MG
RU Santa Mônica	Campus Santa Mônica, Uberlândia, MG
RU Pontal	Campus Pontal, Ituiutaba, MG
RU Glória	Campus Glória, Uberlândia, MG

Fonte – autoria própria

Além da série temporal histórica da quantidade diária de refeições servidas pelos RUs, também foram coletados dados sobre o calendário acadêmico e cardápios, sob a hipótese de que os mesmos exercem influência na demanda de refeições. O período de coleta dos dados está compreendido entre janeiro de 2015 a abril de 2018.

O método empregado neste trabalho é baseado no processo CRISP-DM (CHAPMAN et al., 2000) com adaptações específicas para a realidade de projetos de aprendizado de máquina (SARKAR; BALI; SHARMA, 2017). O fluxograma das atividades executadas está representado na Figura 22. Nas próximas seções as atividades do processo serão apresentadas detalhadamente.

Figura 22 – Fluxograma do processo de elaboração dos modelos de previsão



Fonte – Adaptado de (CHAPMAN et al., 2000) e (SARKAR; BALI; SHARMA, 2017)

5.1 Entendimento do negócio

Nesta fase buscou-se entender as necessidades do RU bem como mapear o problema de negócio para um problema de aprendizado de máquina.

5.1.1 Definição do problema

O RU planeja mensalmente o fornecimento de refeições para o mês subsequente com o objetivo de dimensionar a quantidade de insumos e serviços necessários para atender a esta demanda. A previsão de demanda é uma informação importante para subsidiar este planejamento. Hoje esta previsão é realizada de forma subjetiva pela nutricionista do RU.

O objetivo é elaborar um sistema de previsão quantitativo, baseado em dados históricos de demanda. Outra característica desejada é a de que o modelo construído seja capaz de se atualizar, sem a necessidade de intervenção humana. Tal funcionalidade se torna interessante em uma possível mudança do padrão de demanda em decorrência da expansão de uma unidade, abertura de novos cursos, etc.

Devido à necessidade de planejamento mensal, o horizonte de previsão H foi estabelecido em 30 dias.

5.1.2 Definição da estratégia de aprendizado de máquina

O problema de negócio foi abordado utilizando técnicas de aprendizado de máquina com treinamento supervisionado, onde a saída desejada Y é a quantidade diária de refeições. Os padrões de entrada X serão produzidos nas atividades de engenharia e seleção de

atributos. Dada que a variável dependente (quantidade diária de refeições) pode assumir qualquer valor inteiro, o problema se situa na categoria de tarefas de regressão.

Para se obter a previsão da demanda mensal foi utilizada a estratégia recursiva de múltiplos passos (KLINE, 2003), uma vez que os modelos elaborados são capazes de prever apenas um passo adiante. A previsão mensal será obtida pela soma dos valores obtidos sucessivamente em cada passo, até atingir o horizonte desejado.

Na etapa de construção dos modelos de previsão, foram comparados o desempenho de duas RNAs da família ART: a FAM e EAM. Como linha de base e referência, foram utilizados os algoritmos SNAIVE e KNN.

5.1.3 Identificação das fontes de dados

Na fase de entendimento do negócio buscou-se identificar os fatores que influenciam a demanda de refeições. Verificou-se que o calendário acadêmico e o cardápio são as principais variáveis explicativas que determinam a procura por refeições. Portando, as fontes de dados identificadas foram a página eletrônica da UFU, para se obter o calendário acadêmico, e a página eletrônica do RU para se obter o histórico de cardápios servidos.

A série temporal histórica da demanda diária de refeições foi obtida do banco de dados institucional da UFU.

5.2 Aquisição e entendimento dos dados

Nesta seção serão detalhadas as atividades realizadas para a aquisição e entendimento dos dados utilizados na elaboração dos modelos de previsão.

5.2.1 Quantidade diária de refeições

O controle de acesso aos RUs da UFU é realizado pelo Sistema de Gestão (SG) desenvolvido pelo Centro de Tecnologia da Informação (CTI). A cada entrada de usuário um novo registro é armazenado, em banco de dados relacional, contendo informações como: identificação do usuário, restaurante de origem, modalidade de refeição, data e hora do registro, dentre outras.

A quantidade de refeições servidas diariamente foi obtida pela contagem dos registros agrupando-os por data, modalidade de refeição e restaurante. Foram consideradas apenas

operações de débito (consumo de refeição) e observações a partir do ano de 2015. A consulta SQL utilizada nesta atividade está listada no Código 1.

Código 1 – Consulta SQL submetida para recuperação da movimentação diária dos RUs

```
SELECT
    COUNT(*) as quantidade,
    DATE(data_lancamento) as data,
    id_modalidade_refeicao_ru,
    id_local_ru
FROM
    ufu_movimentacao_ru
WHERE tipo_operacao = 'D' and extract(year from data_lancamento) >= 2015
GROUP BY
    DATE(data_lancamento),
    id_modalidade_refeicao_ru,
    id_local_ru
```

O resultado da consulta foi armazenado em um arquivo no formato CSV cujos campos estão descritos nos Quadros 4 e 5.

Quadro 4 – Descrição dos dados: movimentação diária dos RUs

Campo	**Tipo**	**Descrição**
quantidade	Inteiro	Quantidade de refeições servidas
data	Data	Data da observação
id_modalidade_refeicao_ru	Inteiro	Identificação da modalidade de refeição (ver Quadro 5)
id_local_ru	Inteiro	Identificação do restaurante (ver Quadro 3)
Total de observações	6690	
Data da coleta	24/04/2018	

Fonte – autoria própria

Quadro 5 – Modalidades de refeições servidas nos RUs

Identificador	**Descrição**
1	Café da manhã
2	Almoço
3	Jantar

Fonte – autoria própria

Os dados históricos de demanda diária de refeições foram divididos em dez séries

temporais separadas por restaurante / modalidade de refeição e identificadas conforme Quadro 6. Para cada série foram realizadas as avaliações dos quatro algoritmos (FAM, EAM, KNN e SNAIVE), de forma independente, conforme critérios detalhados posteriormente.

Quadro 6 – Identificação das séries temporais históricas da demanda diária de refeições do restaurante universitário da Universidade Federal de Uberlândia.

Nome	**Descrição**
`GLORIA_ALMOCO`	Série temporal histórica da quantidade diária de refeições servidas no almoço do RU Glória
`PONTAL_ALMOCO`	Série temporal histórica da quantidade diária de refeições servidas no almoço do RU Pontal
`PONTAL_CAFE`	Série temporal histórica da quantidade diária de refeições servidas no café da manhã do RU Pontal
`PONTAL_JANTAR`	Série temporal histórica da quantidade diária de refeições servidas no jantar do RU Pontal
`STAMONICA_ALMOCO`	Série temporal histórica da quantidade diária de refeições servidas no almoço do RU Santa Mônica
`STAMONICA_CAFE`	Série temporal histórica da quantidade diária de refeições servidas no café da manhã do RU Santa Mônica
`STAMONICA_JANTAR`	Série temporal histórica da quantidade diária de refeições servidas no jantar do RU Santa Mônica
`UMU_ALMOCO`	Série temporal histórica da quantidade diária de refeições servidas no almoço do RU Umuarama
`UMU_CAFE`	Série temporal histórica da quantidade diária de refeições servidas no café da manhã do RU Umuarama
`UMU_JANTAR`	Série temporal histórica da quantidade diária de refeições servidas no jantar do RU Umuarama

Fonte – autoria própria

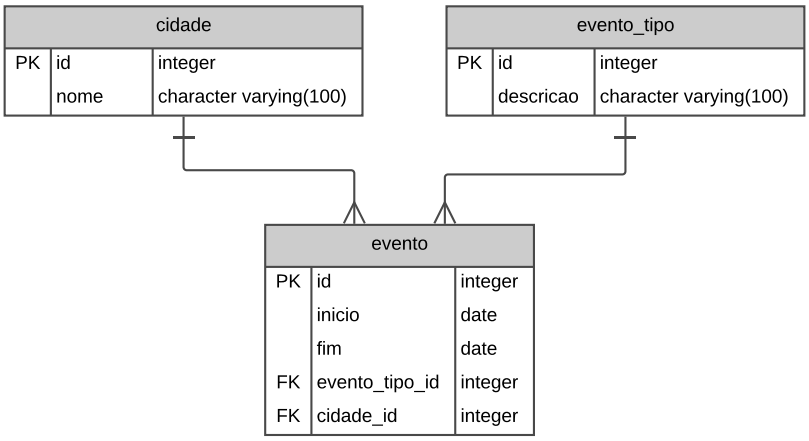
5.2.2 Calendário acadêmico

Eventos como férias acadêmicas, greves, feriados e proximidade a feriados prolongados afetam de forma importante a movimentação dos RUs. Dessa forma, as informações contidas no calendário acadêmico (UFU, 2018c) foram cadastradas em banco de dados relacional para possibilitar a sua utilização na etapa de engenharia de atributos.

A estrutura de tabelas utilizadas está descrita no diagrama entidade-relacionamento ilustrado na Figura 23. Cada evento é composto pela data de início e fim, cidade onde se aplica e o tipo do evento. Devido a diferenças nos calendários (por exemplo feriados municipais) dos campi sediados em Uberlândia (campus Umuarama, campus Santa Mônica

e campus Glória) e do campus Pontal (sediado em Ituiutaba), foi necessário adicionar a informação sobre a cidade. Os possíveis valores para as tabelas `cidade` e `evento_tipo` estão listados nas Quadros 7 e 8 respectivamente. Eventos como recessos ou suspensão das atividades acadêmicas por motivos variados foram todos classificados como tipo 1 (feriado).

Figura 23 – Diagrama entidade-relacionamento das tabelas utilizadas para armazenar o calendário acadêmico



Fonte – autoria própria

Quadro 7 – Possíveis valores para a tabela `cidade`

Identificador	**Cidade**	**Descrição**
1	Todas	Evento se aplica a todas cidades
2	Uberlândia	Evento específico para os campi da cidade de Uberlândia
3	Ituiutaba	Evento específico para os campi da cidade de Ituiutaba

Fonte – autoria própria

Quadro 8 – Possíveis valores para a tabela `evento_tipo`

Identificador	**Tipo do evento**
1	Feriado
2	Férias
3	Greve

Fonte – autoria própria

5.2.3 Cardápio

As informações históricas sobre o cardápio foram coletadas com base na suposição de que o mesmo exerce influência na demanda de refeições dos RUs. Esta hipótese será investigada na fase de seleção de características.

Os dados foram obtidos junto ao setor de *Webmaster* do CTI da UFU, responsável pela manutenção técnica do site do RU (UFU, 2018b). Foi disponibilizado um arquivo no formato CSV compreendendo os cardápios (somente almoço e jantar) oferecidos no período de 08/08/2016 a 01/09/2018 nas quatro unidades do RU. O Quadro 9 descreve os campos contidos no arquivo CSV.

Quadro 9 – Descrição dos dados: cardápio

Campo	**Descrição**
local	Unidade do RU
principal_almoco	Prato principal do almoço
vegetariano_almoco	Prato vegetariano do almoço
arroz_almoco	Tipo de arroz do almoço
feijao_almoco	Tipo de feijão do almoço
guarnicao_almoco	Guarnição do almoço
salada_almoco	Salada do almoço
sobremesa_almoco	Sobremesa do almoço
suco_almoco	Suco do almoço
data	Data da observação
principal_jantar	Prato principal do jantar
vegetariano_jantar	Prato vegetariano do jantar
arroz_jantar	Tipo de arroz do jantar
feijao_jantar	Tipo de feijão do jantar
guarnicao_jantar	Guarnição do jantar
salada_jantar	Salada do jantar
sobremesa_jantar	Sobremesa do jantar
suco_jantar	Suco do jantar
Total de registros	1991
Data da coleta	30/08/2018

5.2.4 Análise exploratória dos dados

Nesta etapa foram utilizadas técnicas de estatística e visualização com objetivo de melhorar o entendimento sobre os dados coletados. As informações aqui obtidas servirão

de subsídio para a escolha das ações mais apropriadas nas atividades subsequentes do processo.

5.2.4.1 Estatística descritiva básica

Nesta atividade os dados foram sumarizados utilizando métricas básicas como a média e desvio padrão. O objetivo é ter uma noção inicial da tendência central dos dados e sua dispersão. Os resultados obtidos podem ser visualizados na Tabela 2.

Tabela 2 – Estatística descritiva básica dos dados históricos da demanda de refeições dos RUs coletados no período de 05/01/2015 a 24/04/2018.

Identificação da série	Quant. de observações	(1)Média	(1)Desvio padrão	(1)Valor mínimo	(1)Valor máximo
UMU_CAFE	451	79.51	25.13	1	129
UMU_ALMOCO	851	763.12	328.59	26	1261
UMU_JANTAR	624	244.97	106.72	1	522
STAMONICA_CAFE	628	109.92	42.30	3	192
STAMONICA_ALMOCO	863	1671.63	712.14	157	2754
STAMONICA_JANTAR	624	893.69	276.41	38	1600
PONTAL_CAFE	713	40.00	25.45	1	100
PONTAL_ALMOCO	925	293.28	183.72	6	738
PONTAL_JANTAR	752	198.42	117.37	3	536
GLORIA_ALMOCO	259	34.25	23.87	1	108

(1) unidade: quantidade de refeições servidas

5.2.4.2 Gráfico de série temporal

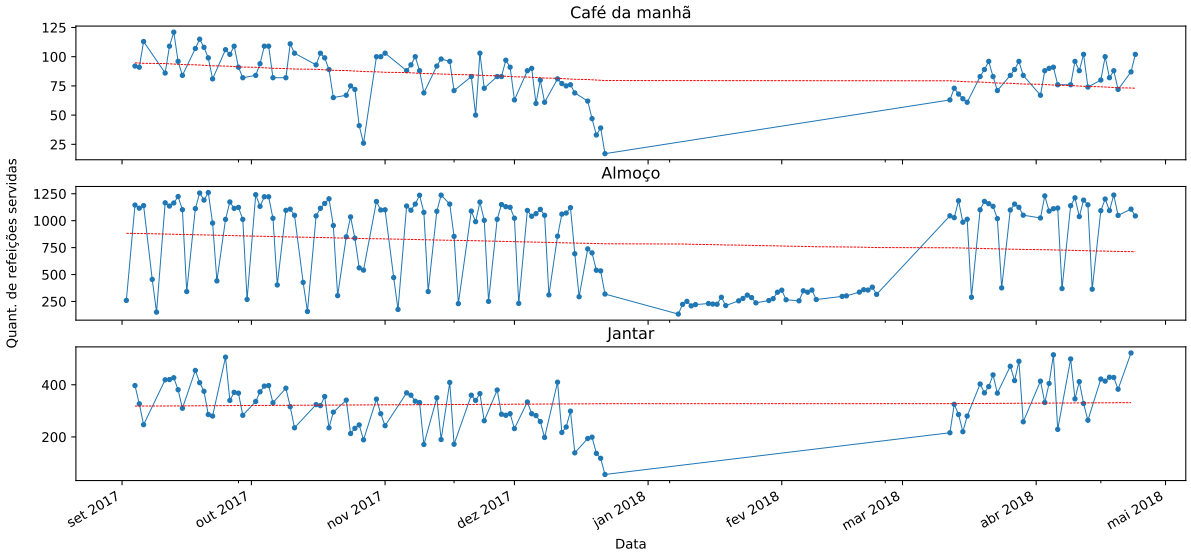
O gráfico de série temporal permite identificar visualmente os componentes básicos da série tais como: tendência, sazonalidade e ciclos (CHATFIELD, 2003). Nos Gráficos 2, 3, 4 e 5 estão representadas as séries temporais da quantidade diária de refeições do RU para os campi Umuarama, Santa Mônica, Pontal e Glória, respectivamente. Para tornar a visualização mais legível, apenas os últimos oito meses foram considerados nos gráficos. No entanto, as análises aqui descritas foram baseadas na visualização completa dos dados.

O componente de tendência foi estimado pelo ajuste de regressão linear simples (PAL; PRAKASH, 2017). O mesmo está representado pela linha vermelha nos gráficos. Foi possível observar uma fraca tendência, seja crescente ou decrescente, em todas as séries.

Por outro lado, observa-se uma forte sazonalidade semanal, que será comprovada pelo estudo da autocorrelação da série.

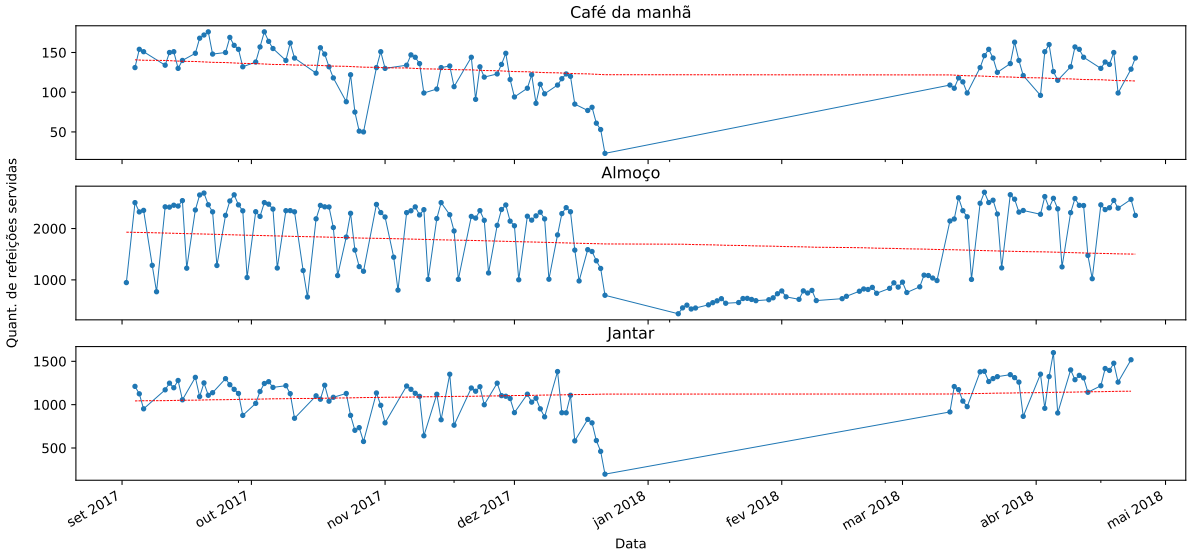
Observaram-se períodos de baixa demanda em decorrência das férias acadêmicas. Estes representam o componente cíclico, pois ocorrem duas vezes ao ano em intervalos mais ou menos regulares. As descontinuidades representadas pelas retas sem observações revelam períodos em que não houve funcionamento do RU, sejam devido às férias, feriados ou recessos.

Gráfico 2 – Gráfico de série temporal da demanda diária de refeições do RU Umuarama. Dados compreendidos entre Setembro de 2017 a Abril de 2018.



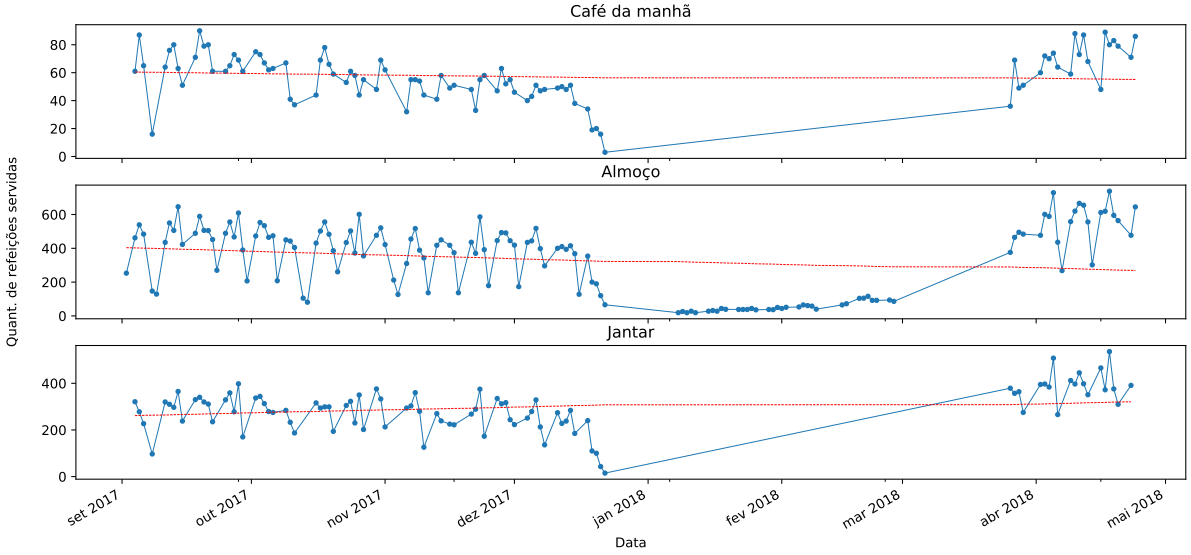
Fonte – autoria própria

Gráfico 3 – Gráfico de série temporal da demanda diária de refeições do RU Santa Mônica. Dados compreendidos entre Setembro de 2017 a Abril de 2018.



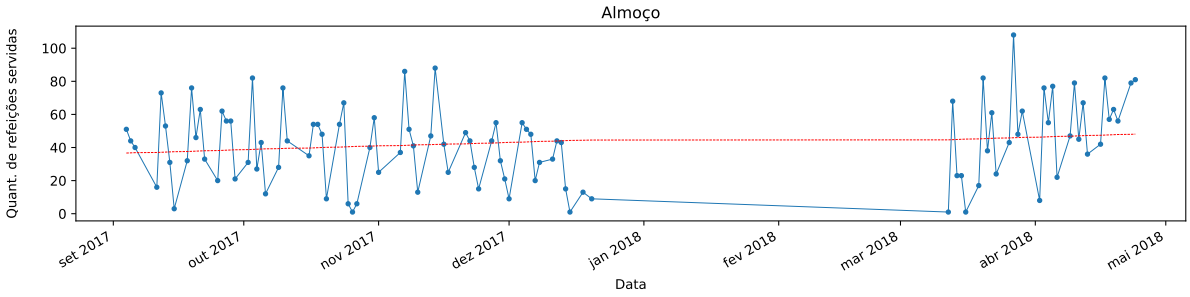
Fonte – autoria própria

Gráfico 4 – Gráfico de série temporal da demanda diária de refeições do RU Pontal. Dados compreendidos entre Setembro de 2017 a Abril de 2018.



Fonte – autoria própria

Gráfico 5 – Gráfico de série temporal da demanda diária de refeições do RU Glória. Dados compreendidos entre Setembro de 2017 a Abril de 2018.



Fonte – autoria própria

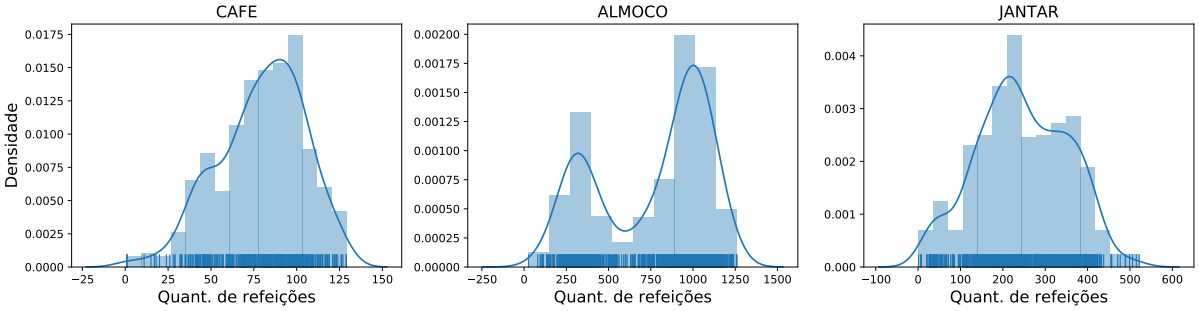
5.2.4.3 Histograma

O histograma é um recurso gráfico que permite identificar com mais detalhes a distribuição dos dados (CHAMBERS et al., 2018). A sua observação pode revelar informações como a tendência central dos dados, dispersão, obliquidade, presença de valores discrepantes e modas (NIST, 2003).

Para os dados históricos da demanda de refeições do RU, foram gerados os histogramas apresentados nos Gráficos 6, 7, 8 e 9, divididos por unidade e modalidade de refeição. Sobrepostas aos histogramas, foram plotadas as curvas da Função densidade de probabilidade (FDP), estimadas com núcleo Gaussiano (HILFIGER, 2015). Observou-se predominantemente uma distribuição bimodal dos dados devido aos períodos de férias acadêmicas e dias letivos, e em alguns casos, multimodais (com três modas).

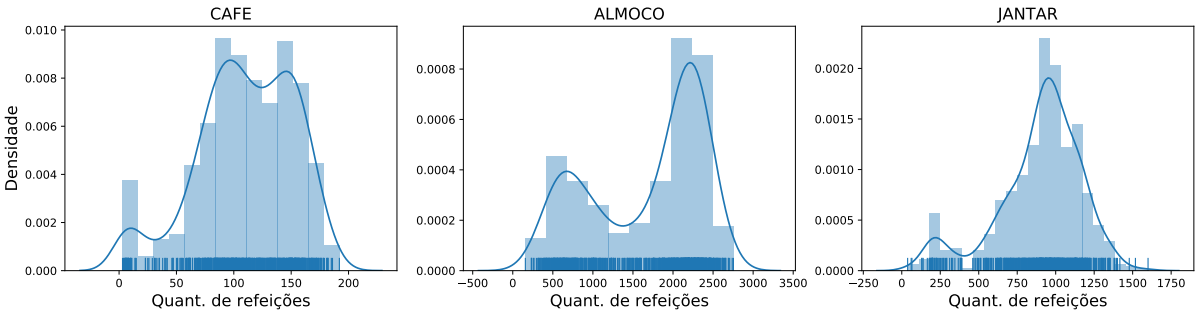
Os gráficos revelaram possível presença de valores discrepantes que podem ser observados nas caldas das distribuições das séries `UMU_CAFE`, `UMU_JANTAR`, `STAMONICA_JANTAR`, `PONTAL_CAFE`, `PONTAL_ALMOCO`, `PONTAL_JANTAR` e `GLORIA_ALMOCO`. No entanto, a análise de valores discrepantes será realizada na próxima seção utilizando o diagrama de caixa.

Gráfico 6 – Histograma da demanda diária de refeições do RU Umuarama. Dados compreendidos entre Janeiro de 2015 a Abril de 2018.



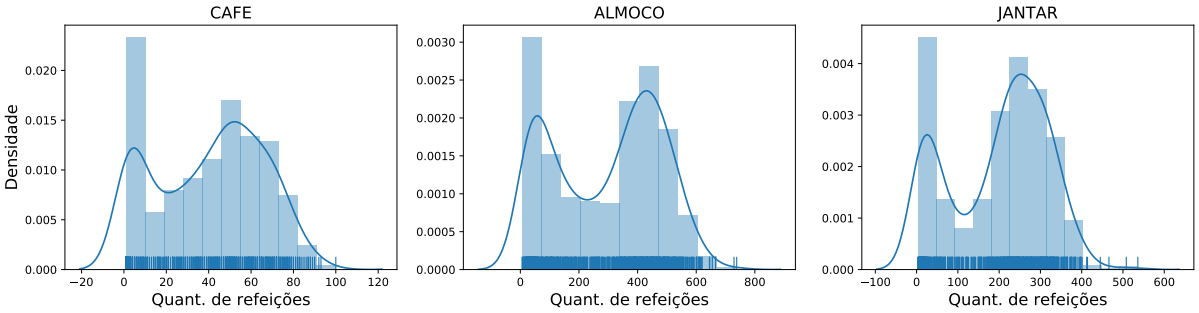
Fonte – autoria própria

Gráfico 7 – Histograma da demanda diária de refeições do RU Santa Mônica. Dados compreendidos entre Janeiro de 2015 a Abril de 2018.



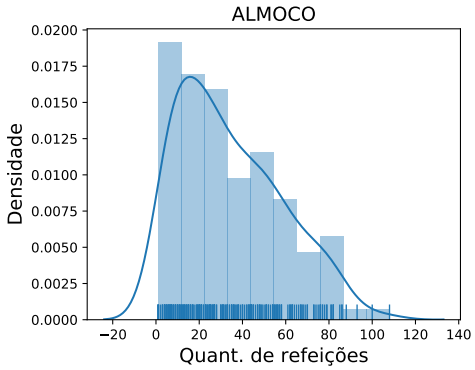
Fonte – autoria própria

Gráfico 8 – Histograma da demanda diária de refeições do RU Pontal. Dados compreendidos entre Janeiro de 2015 a Abril de 2018.



Fonte – autoria própria

Gráfico 9 – Histograma da demanda diária de refeições do RU Glória. Dados compreendidos entre Janeiro de 2015 a Abril de 2018.



Fonte – autoria própria

5.2.4.4 Diagrama de caixa

O diagrama de caixa (*box plot*) também é utilizado para retratar a distribuição dos dados. Ele revela onde estão concentrados 50% dos dados (compreendidos dentro do retângulo), 1º e 3º quartis (arestas inferior e superior do retângulo) e mediana (linha horizontal dentro do retângulo) (CHAMBERS et al., 2018). Nesta pesquisa utilizou-se o diagrama de caixa para o estudo dos valores discrepantes, representados por pontos presentes fora do limite inferior e superior do gráfico.

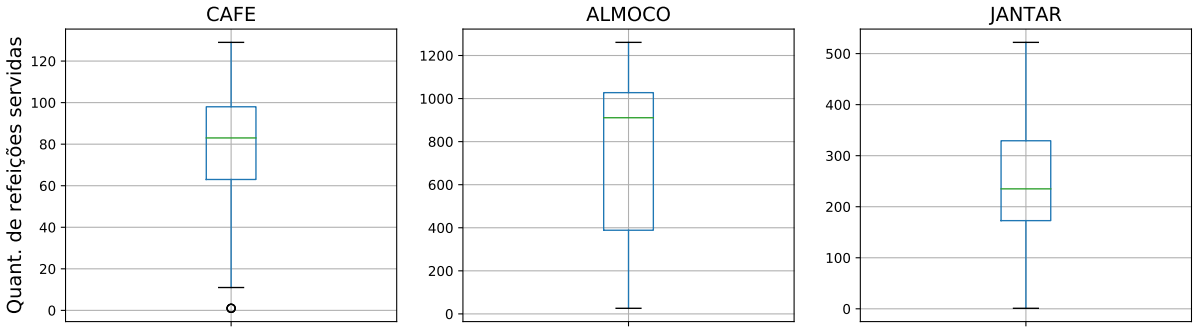
O método indicou valores discrepantes nas séries `UMU_CAFE`, `STAMONICA_JANTAR` e `GLORIA_ALMOCO`, como pode ser constatado nos Gráficos 10, 11 e 13. No entanto, cada caso foi analisado individualmente de forma contextualizada ao calendário acadêmico. Para a série `UMU_CAFE` o método identificou três observações discrepantes com valores iguais a um. Por se tratarem de dias letivos e considerando a baixa frequência de ocorrência, as observações foram consideradas discrepantes.

Para o caso da série `STAMONICA_JANTAR` foram apontadas 41 observações abaixo do limite (com valores entre 38 a 293) e uma observação acima do limite (valor igual a 1600). As observações abaixo do limite não foram consideradas valores discrepantes, pois, ocorrem em períodos de férias, greve ou em semanas adjacentes às férias, justificando a baixa procura por refeições. A observação acima do limite superior (1535) também não foi considerada discrepante, pois, ocorre em período letivo e dista apenas 65 unidades acima.

A única observação (108) apontada como discrepante na série `GLORIA_ALMOCO` não

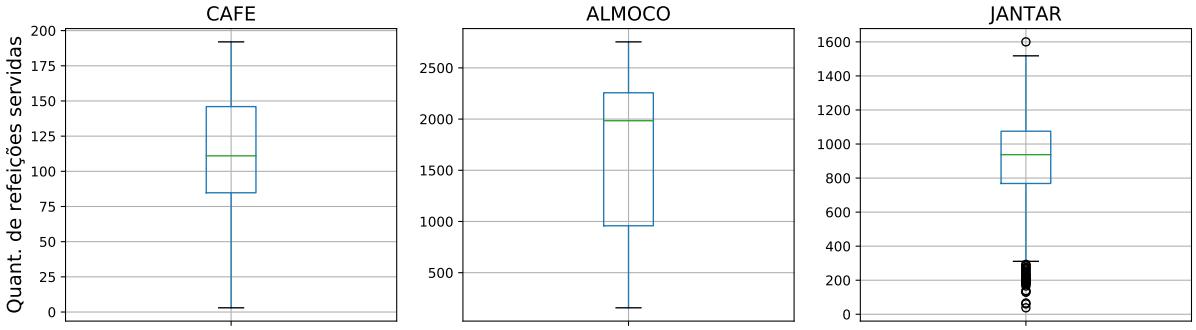
o foi assim classificada, considerando a sua proximidade ao limite superior (106).

Gráfico 10 – Diagrama de caixa da demanda diária de refeições do RU Umuarama. Dados compreendidos entre Janeiro de 2015 a Abril de 2018.



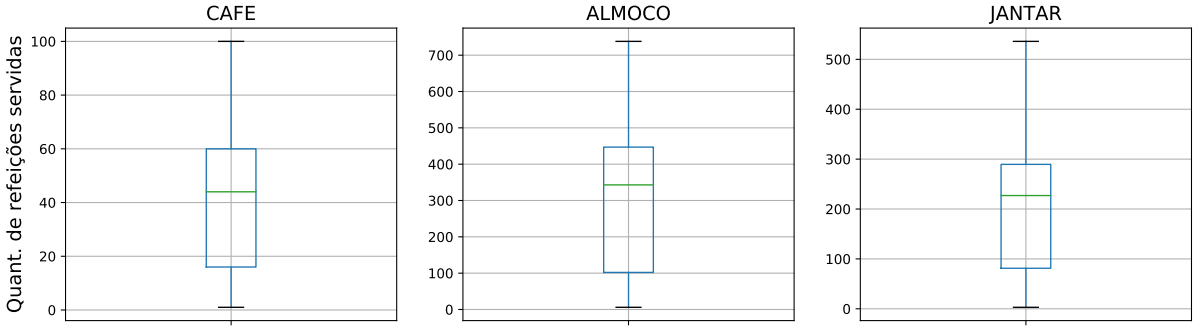
Fonte – autoria própria

Gráfico 11 – Diagrama de caixa da demanda diária de refeições do RU Santa Mônica. Dados compreendidos entre Janeiro de 2015 a Abril de 2018.



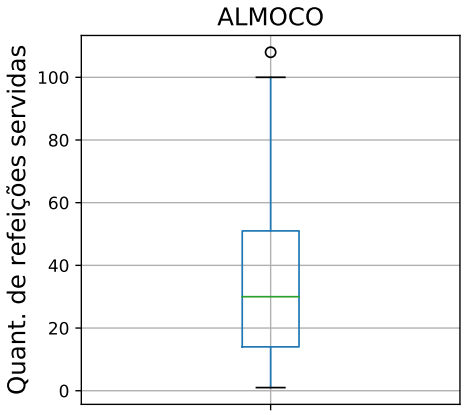
Fonte – autoria própria

Gráfico 12 – Diagrama de caixa da demanda diária de refeições do RU Pontal. Dados compreendidos entre Janeiro de 2015 a Abril de 2018.



Fonte – autoria própria

Gráfico 13 – Diagrama de caixa da demanda diária de refeições do RU Glória. Dados compreendidos entre Janeiro de 2015 a Abril de 2018.



Fonte – autoria própria

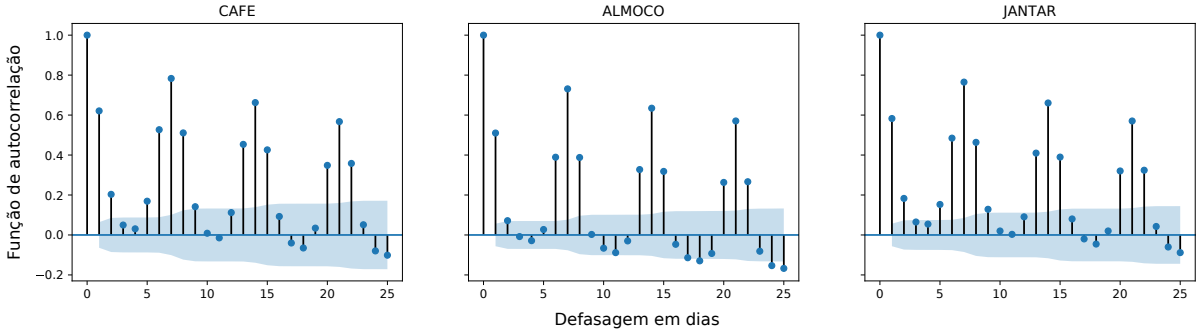
5.2.4.5 Correlograma

O correlograma é um recurso gráfico que auxilia na análise de autocorrelação de uma série temporal. No eixo horizontal são exibidas as defasagens e no eixo vertical o valor do coeficiente de autocorrelação correspondente. Ele pode revelar a presença de tendência caracterizada por coeficientes positivos que decrescem com o aumento das defasagens. Mostra também a sazonalidade da série identificada por picos que se repetem em defasagens múltiplas da frequência de sazonalidade (HYNDMAN; ATHANASOPOULOS, 2018).

A análise dos correlogramas das séries do RU, ilustrados nos Gráficos 14, 15, 16 e 17, revelaram um pico positivo e significantemente diferente de zero na defasagem sete, que se repete em defasagens múltiplas, indicando a sazonalidade semanal das séries. As defasagens um e seis também se mostraram bastante significativas e poderão ser utilizadas como componentes auto-regressivos do modelo de previsão.

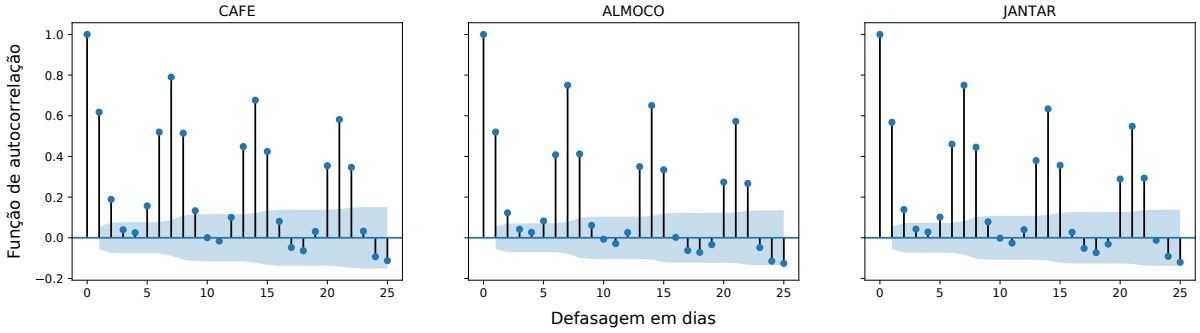
Os componentes auto-regressivos de um modelo, se referem a observações anteriores da própria série temporal que se deseja prever (HYNDMAN; ATHANASOPOULOS, 2018). A análise de correlograma realizada nesta seção, sugere a utilização dos termos passados y_{T-1}, y_{T-6} e y_{T-7} (que serão utilizados como novos atributos) para prever o termo $\hat{y}_T$ da série temporal.

Gráfico 14 – Correlograma da série temporal da demanda diária de refeições do RU Umuarama. Dados compreendidos entre Janeiro de 2015 a Abril de 2018.



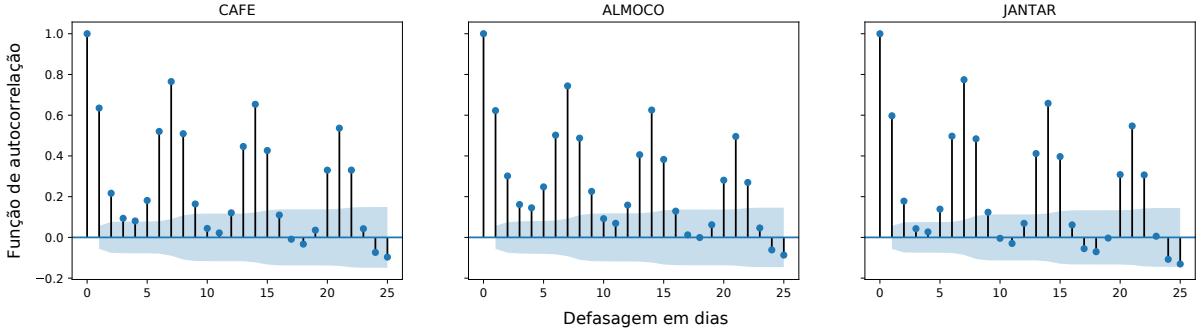
Fonte – autoria própria

Gráfico 15 – Correlograma da série temporal da demanda diária de refeições do RU Santa Mônica. Dados compreendidos entre Janeiro de 2015 a Abril de 2018.



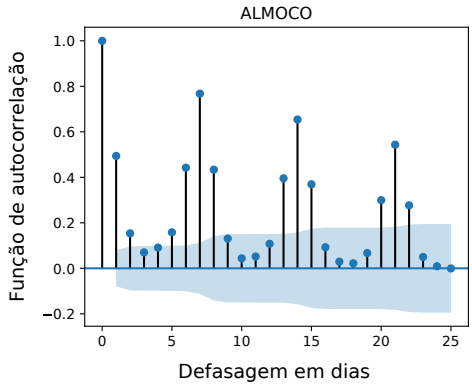
Fonte – autoria própria

Gráfico 16 – Correlograma da série temporal da demanda diária de refeições do RU Pontal. Dados compreendidos entre Janeiro de 2015 a Abril de 2018.



Fonte – autoria própria

Gráfico 17 – Correlograma da série temporal da demanda diária de refeições do RU Glória. Dados compreendidos entre Janeiro de 2015 a Abril de 2018.



Fonte – autoria própria

5.3 Preparação dos dados

Nesta fase do processo os dados são preparados para que possam ser utilizados pelos algoritmos de aprendizado de máquina. As atividades desta etapa compreendem o tratamento de valores discrepantes e inconsistentes, preenchimento de observações ausentes, integração dos dados, engenharia e seleção de atributos.

5.3.1 Pré-processamento dos dados

Com base na análise realizada na seção 5.2.4.4, as observações consideradas discrepantes foram removidas (apenas três observações da série `UMU_CAFE`). Por estarem no início da série, foram consideradas dados oriundos de testes realizados no sistema de controle de acesso, e seus valores não foram substituídos.

Foram constatadas algumas observações cujas datas correspondem a domingo. Estes dados foram considerados como inconsistentes pelo fato de que o RU não fornecer refeições nesse dia. Foram atribuídos a estas observações o valor zero.

Para os demais dias em que não houveram registros de movimentação, foram atribuídos o valor zero, tornando a série igualmente espaçada.

Ainda nesta atividade os dados foram divididos em conjuntos de treinamento e teste. O conjunto de treinamento é utilizado para criar os modelos de previsão e o conjunto de teste para a avaliação final dos modelos. Para cada série, foram reservados os últimos

seis meses para teste, com exceção à série `GLORIA_ALMOCO` que devido ao reduzido número de observações disponíveis (259), apenas os últimos três meses foram utilizados para teste.

Faz-se necessário a divisão dos dados nesse momento com objetivo de se evitar um fenômeno indesejado, muito comum em modelos de previsão: o vazamento de dados. Esta situação é caracterizada quando se utiliza informações que não são legitimamente disponíveis para ajustar de avaliar os modelos. O principal sintoma da ocorrência do vazamento de dados é a criação de modelos cuja desempenho são excessivamente otimistas em tempo de projeto, mas que apresentam baixa capacidade de generalização ao serem implantados em ambiente de produção (KAUFMAN et al., 2012).

5.3.2 Engenharia de atributos

No contexto de aprendizado de máquina, atributos ou *features* são informações obtidas dos dados brutos, comumente representados por valores numéricos. Por dados brutos entende-se por informações coletadas diretamente das fontes de dados, que passaram pela atividade de pré-processamento, no entanto, ainda não estão em um formato aceitável para os algoritmos de aprendizado de máquina. O processo de criação de atributos é denominado de engenharia de atributos ou *feature engineering*. A criação e identificação dos atributos corretos determinam a qualidade dos modelos preditivos na medida em que facilitam a identificação do relacionamento entre as entradas e saídas pelos algoritmos baseados em treinamento supervisionado (SARKAR; BALI; SHARMA, 2017).

Serão apresentadas nas próximas seções os tipos de atributos criados para o problema em questão.

5.3.2.1 Atributos de data

São atributos obtidos diretamente da data de ocorrência da observação. Com base no estudo da sazonalidade das séries, realizado na seção 5.2.4.5, verificou-se que o dia da semana exerce influência sob a demanda de refeições. Para representar esta informação, foi criado o atributo dia da semana ($\mathtt{a_d DIA\_SEMANA}$) que pode assumir valores inteiros do intervalo $[0, 6]$ onde o primeiro valor corresponde à segunda-feira e o último à domingo.

De forma análoga foi criado o atributo mês ($\mathtt{a_d MES}$) para representar o mês da ocorrência da observação. O mesmo pode assumir valores inteiros do intervalo $[0, 11]$.

Para captar uma possível mudança nos padrões de movimentação do RU em decorrência dos recessos que ocorrem no fim e início de ano foi criado o atributo da primeira e última semana do ano (a_dPUSA). Este atributo assume o valor 1 caso a semana da observação seja a primeira ou a última semana do ano e zero para os demais casos.

5.3.2.2 Atributos de calendário

Verificou-se na atividade de análise exploratória dos dados que o calendário acadêmico exerce influência sobre a demanda de refeições do RU. Eventos como férias, feriados e recessos são importantes variáveis explicativas para os modelos de previsão. Além dos eventos explícitos no calendário, informações como véspera de férias, véspera de feriado prolongado e início de semestre também podem representar atributos relevantes.

Uma vês que o calendário acadêmico foi cadastrado em um banco de dados relacional conforme descrito na seção 5.2.2, a criação dos atributos foi viabilizada através da consulta SQL listada no Apêndice A. O Quadro 10 apresenta os atributos gerados nesta atividade.

Quadro 10 – Atributos de calendário

Atributo	**Descrição**
a_cFERIADO	atributo feriado
a_cFERIAS	atributo férias
a_cGREVE	atributo greve
a_cVESPERA_FERIAS	atributo das duas semanas que antecedem as férias
a_cPRIMEIRA_SEMANA	atributo da primeira semana do semestre
a_cVESPERA_FERIADO_PROLONGADO	atributo dos dois dias que antecedem um feriado prolongado
a_cSEGUNDA_VESPERA_FERIADO	atributo da segunda-feira véspera de feriado
a_cSEXTA_POS_FERIADO	atributo da sexta-feira subsequente a um feriado

Fonte – autoria própria

5.3.2.3 Atributos de janela deslizante

Os atributos de janela deslizante são criados pela utilização de funções de agregação cujas entradas são W observações precedentes à observação que se deseja criar o atributo correspondente, onde W é o tamanho da janela. Para tornar o procedimento mais claro, considere a série formada pelas observações $y_1, y_2, \ldots, y_t$. Suponha que se deseja calcular

o atributo de janela deslizante $\mathrm{a_{jd}}F_n(y_t)$ correspondente à observação y_t onde F_n é uma função de agregação. O valor de $\mathrm{a_{jd}}F_n(y_t)$ será dado por (5.1).

$$\mathrm{a_{jd}}F_n(y_t) = F_n(y_{t-1}, y_{t-2}, \ldots, y_{t-W}) \tag{5.1}$$

A função F_n pode assumir qualquer procedimento que de alguma forma agregue as entradas retornando apenas um valor numérico, por exemplo: média, desvio padrão, variância, soma, valor mínimo, valor máximo, etc.

Para o problema abordado nesse trabalho, o tamanho da janela deslizante W foi fixado em sete, considerando o estudo de autocorrelação apresentado na seção 5.2.4.5. Foram utilizadas 43 funções de agregação implementadas no pacote *Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests* (tsfresh) (CHRIST et al., 2018) disponível para a linguagem de programação Python. Algumas dessas funções podem receber parâmetros adicionais que alteram a sua resposta, por exemplo, a função `value_count(x, value)` que conta a quantidade de ocorrências de `value` no vetor `x`. Ao todo foram criados 93 atributos de janela deslizante e serão identificados neste trabalho no seguinte formato: $\mathtt{a_{jd}}$`NOME_DO_ATRIBUTO`.

5.3.2.4 Atributo do cardápio

O objetivo desta atividade foi associar uma categoria numérica a cada tipo diferente de cardápio. Apenas o componente prato principal do cardápio foi considerado. Esta informação é do tipo cadeia de caracteres ou *string* sem nenhum tipo de padronização, preenchido por um funcionário do RU de forma livre. Ou seja, o mesmo prato pode ser representado por diferentes variações de *strings*.

O primeiro passo foi criar uma lista única de pratos principais oriundos do almoço e jantar, de todas as observações obtidas na seção 5.2.3, e então retiradas as ocorrências repetidas (*strings* iguais). Através de uma inspeção visual houveram apenas duas ocorrências em que o mesmo prato estava com a grafia diferente, os quais foram retirados da lista. Para representar os dias em que não houveram fornecimento de refeições, foi adicionado à lista a *string* “sem fornecimento de refeicao”, obtendo-se assim 103 categorias, identificadas de 0 a 102.

Uma vez identificadas as categorias de cardápio, os modelos de previsão podem ser treinados com esta informação visto que o atributo foi mapeado a uma grandeza numérica.

No entanto, eventualmente, o modelo poderá ser atualizado com novas observações ou utilizado para realizar previsões. Desta forma, é necessário mapear a informação de cardápio "não vista" (que não foi utilizada no momento do treinamento) em umas das categorias identificadas no passo anterior. Dado o problema da falta de padronização exposto no início da seção, uma medida de similaridade deve ser adotada.

Para tratar este problema foi utilizado o conceito MEV, onde cada documento (no caso, cada descrição de prato principal) é representado por um vetor numérico. Neste trabalho foi utilizado o modelo TF-IDF para vetorizar os documentos (mais detalhes em 2.2.4.2.3). Ao receber uma nova informação a mesma técnica é aplicada, obtendo-se um vetor correspondente. Então é realizado um cálculo de similaridade (neste trabalho foi utilizada a distância Euclidiana) entre o novo vetor e os vetores já conhecidos, encontrando assim a categoria mais apropriada para a nova observação (SARKAR, 2016). Este atributo será identificado como $\mathtt{a}_{\mathtt{card}}$.

5.3.3 Seleção de atributos

Na etapa anterior foram obtidos três atributos de data, oito atributos de calendário, 93 atributos de janela deslizante e um atributo de cardápio, totalizando 105 atributos. No entanto, de uma grande quantidade de atributos, podem decorrer diversos fenômenos indesejados mais conhecidos como a "maldição da dimensionalidade" (BISHOP, 2006).

A seleção de atributos busca identificar a quantidade ótima de atributos com o objetivo de elaborar modelos com boa capacidade de generalização, reduzir o custo computacional, evitar o sobreajuste e ainda proporciona um melhor entendimento sobre a importância dos atributos (SARKAR; BALI; SHARMA, 2017).

A abordagem utilizada para o problema de seleção de atributos foi a utilização de dois métodos. O primeiro, dentro da categoria dos métodos de filtro, elimina os atributos altamente correlacionados. Uma matriz é criada onde o coeficiente de correlação de Pearson (MAINDONALD; BRAUN, 2010) é calculado para todos os possíveis pares de atributos. Para os pares onde o valor absoluto do coeficiente de correlação é superior a $0,95$, o atributo acima da diagonal principal é removido.

No segundo passo foi utilizada a técnica de Eliminação recursiva de atributos com validação cruzada (ERAVC). Nesta técnica um estimador externo é utilizado para atribuir pesos aos atributos. O procedimento se inicia utilizando todo o conjunto de atributos e

aqueles de menor importância são eliminados (a quantidade de atributos a serem eliminados por iteração é escolhida previamente). O procedimento se repete até se obter a quantidade desejada de atributos. Utilizando a técnica de validação cruzada, além da seleção dos melhores atributos, é possível se obter a quantidade ideal dos mesmos (ALBON, 2018). Neste trabalho foi utilizado o estimador Floresta Aleatória para atribuição dos pesos aos atributos (BREIMAN, 2001). O Quadro 11 lista o resultado da fase de seleção de atributos, realizada para cada série temporal.

Quadro 11 – Resultado da etapa de seleção de atributos

(continua)

Série	**Atributos selecionados**	**Quant.**
GLORIA_ALMOCO	a_{jd}CWT_COEFFICIENTS__W_2, a_{jd}LAG7, a_{jd}TIME_REVERSAL_ASYMMETRY_TATISTIC__LAG_2, a_{card}, a_dDIA_SEMANA, a_cFERIADO, a_cFERIAS, a_cGREVE, a_cVESPERA_FERIAS	9
PONTAL_ALMOCO	a_{jd}ABS_ENERGY, a_{jd}CWT_COEFFICIENTS__W_2, a_{jd}LAG1, a_{jd}LAG6, a_{jd}TIME_REV_ASYMM_STAT__LAG_1, a_{card}, a_dDIA_SEMANA, a_cFERIADO, a_cFERIAS, a_cGREVE, a_cVESPERA_FERIAS	11
PONTAL_CAFE	a_{jd}ABS_ENERGY, a_{jd}ABSOLUTE_SUM_OF_CHANGES, a_{jd}CWT_COEFFICIENTS__W_2, a_{jd}LAG1, a_{jd}LAG7, a_{jd}TIME_REV_ASYMM_STAT__LAG_1, a_dDIA_SEMANA, a_cFERIADO, a_cFERIAS, a_cGREVE, a_cVESPERA_FERIAS	11
PONTAL_JANTAR	a_{jd}ABS_ENERGY, a_{jd}ABSOLUTE_SUM_OF_CHANGES, a_{jd}CWT_COEFFICIENTS__W_10, a_{jd}CWT_COEFFICIENTS__W_2, a_{jd}LAG1, a_{jd}TIME_REV_ASYMM_STAT__LAG_1, a_{card}, a_dDIA_SEMANA, a_dMES, a_cFERIADO, a_cFERIAS, a_cGREVE, a_cVESPERA_FERIAS, a_cPRIMEIRA_SEMANA	14
STAMONICA_ALMOCO	a_{jd}ABSOLUTE_SUM_OF_CHANGES, a_{jd}LAG1, a_{jd}LAG6, a_{jd}LAG7, a_{jd}MAXIMUM, a_{jd}QUANTILE__Q_0.3, a_{jd}STANDARD_DEVIATION, a_{jd}TIME_REV_ASYMM_STAT__LAG_3, a_{card}, a_dDIA_SEMANA, a_dMES, a_cFERIADO, a_cFERIAS, a_cGREVE, a_cVESPERA_FERIAS	15
STAMONICA_CAFE	a_{jd}FIRST_LOCATION_OF_MAXIMUM, a_{jd}LAG1, a_{jd}LAG7, a_{jd}MAXIMUM, a_dDIA_SEMANA, a_cFERIADO, a_cFERIAS, a_cGREVE, a_cVESPERA_FERIAS	9

Quadro 11 – Resultado da etapa de seleção de atributos

(conclusão)

Série	**Atributos selecionados**	**Quant.**
STAMONICA_JANTAR	a_{jd}ABS_ENERGY, a_{jd}ABSOLUTE_SUM_OF_CHANGES, a_{jd}AUTOCORRELATION__F_AGG_MEDIAN, a_{jd}AUTOCORRELATION__F_AGG_VAR, a_{jd}BINNED_ENTROPY__MAX_BINS_10, a_{jd}CWT_COEFFICIENTS__W_10, a_{jd}CWT_COEFFICIENTS__W_2, a_{jd}LAG1, a_{jd}LAG2, a_{card}, a_{d}DIA_SEMANA, a_{d}MES, a_{c}FERIADO, a_{c}FERIAS, a_{c}GREVE, a_{c}VESPERA_FERIAS	16
UMU_ALMOCO	a_{jd}ABS_ENERGY, a_{jd}CWT_COEFFICIENTS__W_2, a_{jd}LAG1, a_{card}, a_{d}DIA_SEMANA, a_{d}MES, a_{c}FERIADO, a_{c}FERIAS, a_{c}GREVE, a_{c}VESPERA_FERIAS	10
UMU_CAFE	a_{jd}ABS_ENERGY, a_{jd}CWT_COEFFICIENTS__W_2, a_{jd}FIRST_LOCATION_OF_MAXIMUM, a_{jd}LAG1, a_{jd}LAG7, a_{jd}LINEAR_TREND__ATTR_RVALUE, a_{jd}MAXIMUM, a_{d}DIA_SEMANA, a_{c}FERIADO, a_{c}FERIAS, a_{c}GREVE, a_{c}VESPERA_FERIAS	12
UMU_JANTAR	a_{jd}ABS_ENERGY, a_{jd}AUTOCORRELATION__F_AGG_MEAN, a_{jd}SUM_OF_REOCCURRING_DATA_POINTS, a_{d}DIA_SEMANA, a_{d}MES, a_{c}FERIADO, a_{c}FERIAS, a_{c}GREVE, a_{c}VESPERA_FERIAS, a_{c}PRIMEIRA_SEMANA, a_{c}VESPERA_FERIADO_PROLONGADO, a_{c}SEXTA_POS_FERIADO	12

Fonte – autoria própria

A seleção de atributos descrita nesta seção foi empregada para o treinamento dos algoritmos KNN, FAM e EAM. Para o método SNAIVE, buscou-se tornar o processo de modelagem o mais simples possível. Com isso um modelo inicial é construído com um esforço reduzido servindo de linha de base para os métodos mais complexos. Pela mesma razão a seleção de atributos para o modelo SNAIVE baseou-se no princípio em considerar os atributos mais simples de serem obtidos e escolhidos aqueles mais importantes, baseado no entendimento do negócio em questão. Assim, os atributos selecionados para o modelo SNAIVE foram : a_{d}DIA_SEMANA, a_{c}FERIADO e a_{c}FERIAS.

5.3.4 Transformação de atributos

Dependendo do algoritmo de aprendizado de máquina utilizado, alguns ajustes na magnitude dos atributos são necessários. A RNA FAM requer que os atributos e a resposta desejada estejam normalizados no intervalo $[0, 1]$. Esta transformação foi obtida utilizando a equação (5.2).

$$x'_i = \frac{x_i - min(x)}{max(x) - min(x)} \tag{5.2}$$

onde $max(x)$ e $min(x)$ são respectivamente os valores máximo e mínimo que a variável x pode assumir e x'_i é o valor correspondente à transformação de x_i. Este tipo de transformação assume a existência de um valor máximo e de um valor mínimo para o atributo. No entanto, considerando que os modelos serão atualizados de forma contínua, esta restrição pode ser violada. Para lidar com este problema, foi utilizado um algoritmo capaz de calcular de forma incremental a normalização dos dados.

Algoritmos como KNN e EAM utilizam distância euclidiana na sua implementação. Se um dos atributos possuir uma escala grande, ele terá uma influência maior no cálculo da distância se comparado a outro atributo com uma escala menor, gerando resultados distorcidos. Para contornar esta situação os atributos foram transformados utilizando o escore padronizado. Neste procedimento todos os atributos terão média zero e desvio padrão um. Desta forma os dados estarão em uma mesma escala, mais apropriada para comparações (JAMES et al., 2017).

Os atributos $\mathtt{a_d DIA\_SEMANA}$, $\mathtt{a_c FERIADO}$, $\mathtt{a_c FERIAS}$, $\mathtt{a_c GREVE}$, $\mathtt{a_c VESPERA\_FERIAS}$, $\mathtt{a_c PRIMEIRA\_SEMANA}$, $\mathtt{a_c VESPERA\_FERIADO\_PROLONGADO}$, $\mathtt{a_c SEGUNDA\_VESPERA\_FERIADO}$ e $\mathtt{a_c SEXTA\_POS\_FERIADO}$, apesar de possuírem a sua representação numérica, necessitam de mais um nível de transformação. Por serem atributos categóricos deseja-se que os modelos assimilem a relação de classe ou agrupamento dos dados e não uma relação de ordinalidade. Para tal foi utilizado a transformação *one hot encoding*. O procedimento cria um atributo para cada categoria que assume o valor um na presença da categoria ou zero caso contrário (SARKAR; BALI; SHARMA, 2017). O Quadro 12 exemplifica a técnica aplicada para o atributo $\mathtt{a_d DIA\_SEMANA}$.

O atribulo $\mathtt{a_{card}}$ também é categórico, no entanto, devido à quantidade elevada de categorias (103) a codificação *one hot encoding* torna-se inapropriada. Para este caso foi

Quadro 12 – Codificação *one hot encoding* para o atributo $\mathtt{a_d DIA\_SEMANA}$

		one hot encoding						
Categoria	**Numérico**	**cat$_1$**	**cat$_2$**	**cat$_3$**	**cat$_4$**	**cat$_5$**	**cat$_6$**	**cat$_7$**
Segunda	0	1	0	0	0	0	0	0
Terça	1	0	1	0	0	0	0	0
Quarta	2	0	0	1	0	0	0	0
Quinta	3	0	0	0	1	0	0	0
Sexta	4	0	0	0	0	1	0	0
Sábado	5	0	0	0	0	0	1	0
Domingo	6	0	0	0	0	0	0	1

Fonte – autoria própria

adotada a codificação *feature hashing*. Nesta técnica cada categoria é transformada em um vetor de dimensão preestabelecida. No entanto, como a dimensão do vetor é menor que o número de categorias, podem ocorrer colisões, onde diferentes categorias são mapeadas em um mesmo vetor (SARKAR; BALI; SHARMA, 2017). O Quadro 13 ilustra o resultado da codificação para uma amosta de cinco categorias do atributo $\mathtt{a_{card}}$.

Quadro 13 – Codificação *feature hashing* para o atributo $\mathtt{a_{card}}$

		feature hashing				
Categoria	**Numérico**	**$\mathtt{a_{card}}$0**	**$\mathtt{a_{card}}$1**	**$\mathtt{a_{card}}$2**	**$\mathtt{a_{card}}$3**	**$\mathtt{a_{card}}$4**
Galinhada	60	8	1	-6	-2	-3
Lasanha à Bolonhesa	73	12	0	-8	-1	-3
Estrogonofe de Frango	36	9	3	-6	1	1
Filé de Peixe Assado ao Molho de Moqueca	43	5	2	-5	0	-6
Lagarto ao Molho Madeira	69	12	3	-4	1	-4

Fonte – autoria própria

5.4 Validação da implementação das RNA *Fuzzy ARTMAP* e *Euclidean ARTMAP*

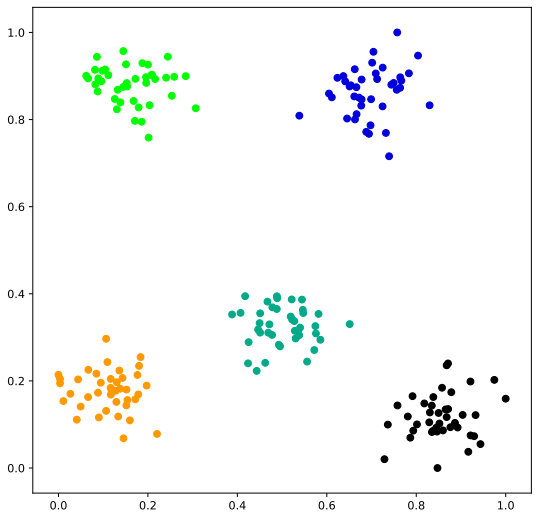
Antes de dar continuidade ao processo de modelagem da série temporal da demanda diária de refeições do RU será detalhado nas próximas seções o procedimento de homologação que foi realizado com o objetivo de assegurar o funcionamento correto das RNAs FAM e EAM, uma vez que foram obtidas por implementação própria.

5.4.1 Capacidade de clusterização

Cada uma das RNAs FAM e EAM são elaboradas utilizando dois módulos das redes FA (CARPENTER; GROSSBERG; ROSEN, 1991b) e EA (KENAYA; CHEOK, 2008a) respectivamente. Estas redes por sua vez possuem treinamento não supervisionado e são utilizadas para clusterização de dados. Nesta atividade os módulos FA e EA foram testados antes de serem utilizados nas redes FAM e EAM.

Inicialmente foi gerado de forma aleatória um conjunto de dados contendo 200 pontos no plano, aglomerados em cinco centros, conforme ilustrado na Figura 24.

Figura 24 – Dados utilizados para teste de clusterização das redes FA e EA.



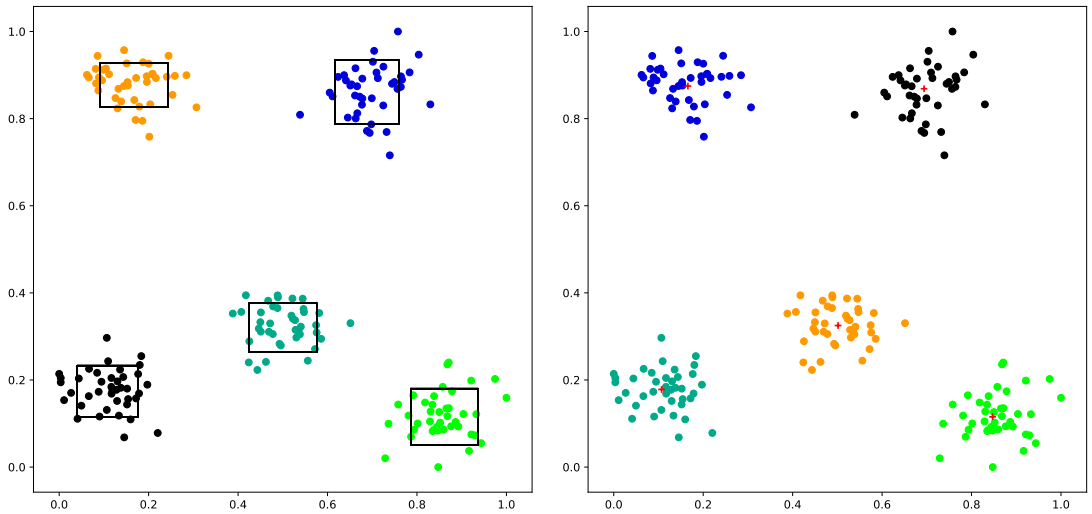
Fonte – autoria própria

Posteriormente, as duas redes foram treinadas com os mesmos dados. O resultado da clusterização está representado na Figura 25. A formação dos agrupamentos é indicada pelos pontos de mesma cor (não necessariamente iguais à cores dos dados de treinamento). O resultado da clusterização pela rede FA pode ser verificado na Figura 25(a). Os retângulos são a representação geométrica das categorias de agrupamento, criadas no processo de treinamento. Da mesma forma a Figura 25(b) representa o resultado da clusterização da rede EA. As centróides estão representadas pelo sinal de adição, em vermelho.

Nesta avaliação ambas as redes obtiveram um acerto de 100%, agrupando corretamente os dados nas classes originais. Uma vez que as redes FA e EA apresentaram o comportamento desejado, foram empregadas para a implementação das redes FAM e EAM.

Figura 25 – Resultado do teste de clusterização das redes FA e EA.

(a) Clusterização FA (b) Clusterização EA



Fonte – autoria própria

5.4.2 Aproximação de função

O teste das redes FAM e EAM, de treinamento supervisionado, compreendeu em avaliar a capacidade de aproximação da função (5.3).

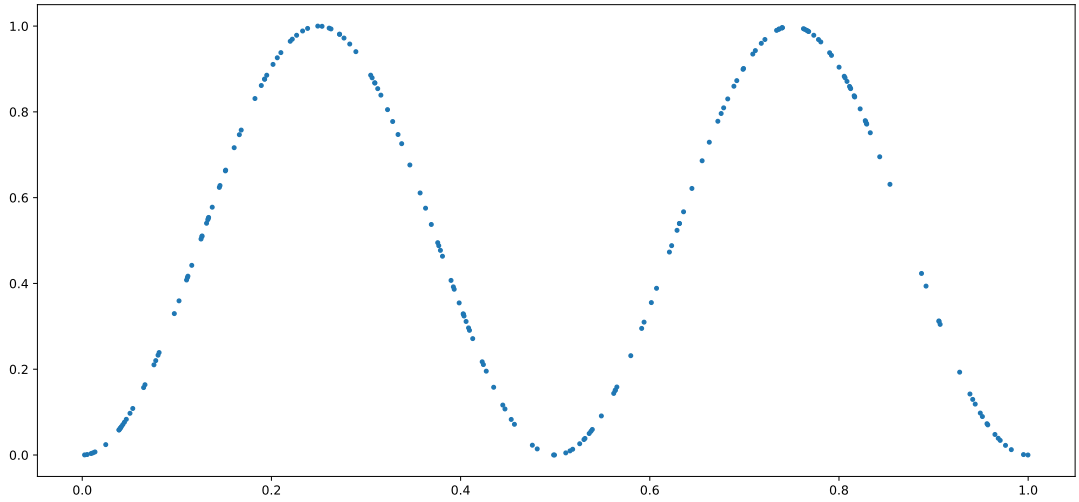
$$y = f(x) = (\sin 2\pi x)^2 \tag{5.3}$$

onde y é a saída desejada (variável dependente).

Para a formação do conjunto de treinamento foram escolhidos de forma aleatória 200 valores para $x \in [0, 1]$. A Figura 26 representa o conjunto de pares $(x, f(x))$ obtidos. Exibidas na Figura 27 estão representadas as categorias formadas no processo de treinamento da rede FAM.

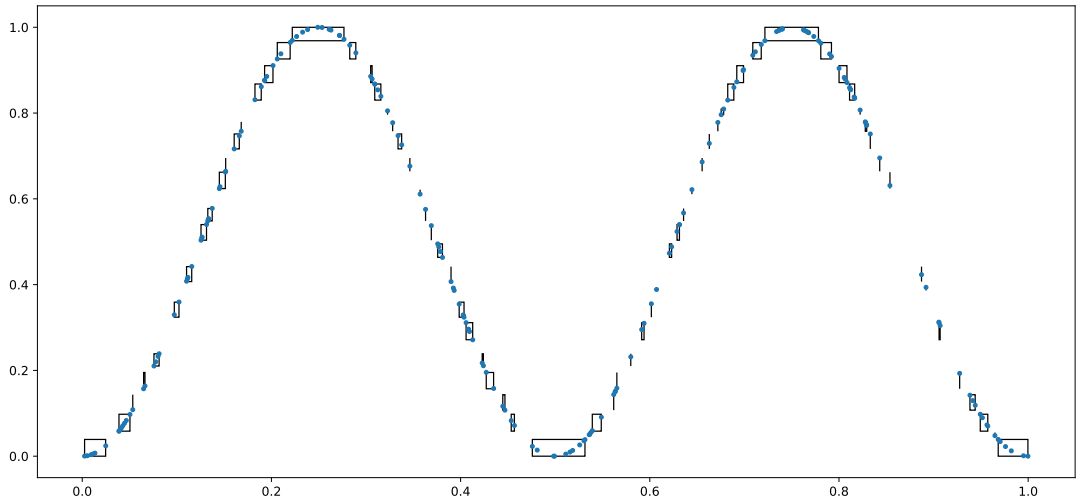
Posteriormente os modelos foram testados utilizando outro conjunto de dados composto por 200 pares $(x, f(x))$, tal que $x \in [0, 1]$ escolhidos aleatoriamente. O resultado da aproximação dos modelos podem ser observados na Figura 28. Os pontos em azul representam os dados de teste e a curva laranja a saída do modelo. Para quantificar o erro cometido pelos modelos, foi utilizada a métrica MAE. Como pode ser observado graficamente e através dos resultados das métricas, os modelos conseguiram se ajustar corretamente aos dados de treinamento, respondendo satisfatoriamente no procedimento de teste.

Figura 26 – Dados utilizados para treinamento das redes FAM e EAM.



Fonte – autoria própria

Figura 27 – Representação geométrica das categorias formadas no processo de treinamento da rede FAM. Hiperparâmetros: $\alpha = 0,001$, $\beta = 0,99$ e $p_b = 0,96$. Nº de categorias $\text{FA}_a : 75$, $\text{FA}_b : 25$.

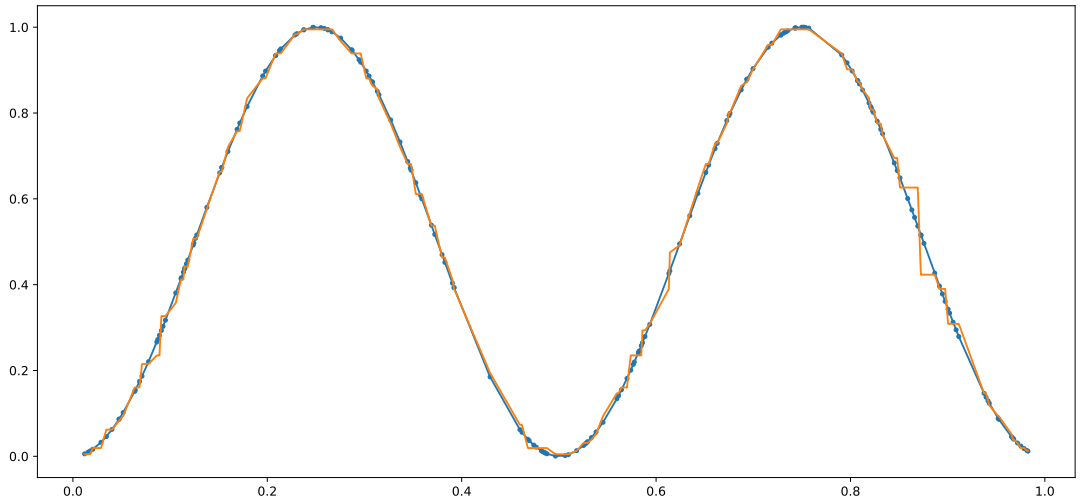


Fonte – autoria própria

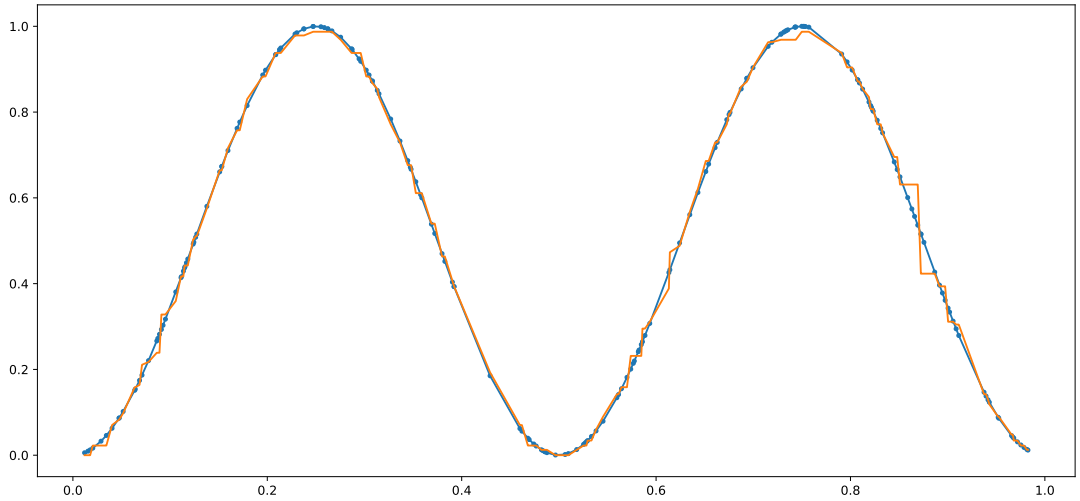
Nas próximas seções é dada continuidade ao processo de modelagem da série temporal do RU.

Figura 28 – Resultado do teste de aproximação de função das redes FAM e EAM.

(a) Resultado da rede FAM. MAE = 0,0116.



(b) Resultado da rede EAM. MAE = 0,0123.



Fonte – autoria própria

5.5 Modelagem e avaliação

Nesta fase do processo o objetivo foi estimar os hiperparâmetros dos modelos utilizando o conjunto de treinamento. Posteriormente cada modelo será avaliado conforme critérios abordados nas próximas seções.

5.5.1 Ajuste de hiperparâmetros

O ajuste de hiperparâmetros foi realizado utilizando a técnica de busca aleatória ou *random search*. Inicialmente defini-se a quantidade de iterações que se deseja executar.

Posteriormente, para cada hiperparâmetro é definido um espaço de busca de onde os valores serão escolhidos aleatoriamente. Este espaço de busca pode ser definido por uma lista de valores ou por uma distribuição de probabilidade. A cada iteração um modelo candidato é criado e avaliado em regime de validação cruzada. Aquele que apresentar o melhor desempenho é escolhido (SARKAR; BALI; SHARMA, 2017).

Devido à dependência temporal dos dados (ocorrem de forma sucessiva), um cuidado especial deve ser tomado no procedimento de validação cruzada. Deve-se garantir que observações futuras não sejam utilizadas no treinamento do modelo evitando assim uma avaliação tendenciosa. Para contornar esta situação foi utilizado o procedimento de validação cruzada para séries temporais, denominado *rolling origin* (HYNDMAN; ATHANASOPOULOS, 2018). Neste trabalho o conjunto de treinamento foi particionado em 12 dobras, com horizonte de previsão igual a 20 dias (conjunto de validação). A métrica de avaliação utilizada no procedimento de validação cruzada foi a RMSE.

O procedimento foi repetido para cada modelo utilizando o conjunto de treinamento de cada série temporal. Não foi necessário aplicar o método para o modelo SNAIVE, pois o mesmo não possui hiperparâmetros para serem ajustados. Em cada execução do procedimento, 200 candidatos foram avaliados. Para o modelo KNN, todos os possíveis valores para o hiperparâmetro k foram testados, uma vez que o mesmo é um número inteiro obtido no intervalo $[1, 15]$. O Quadro 14 mostra a configuração do espaço de busca para os hiperparâmetros dos modelos. Como pode ser observado nos modelos FAM e EAM, apenas o parâmetro de taxa de aprendizagem β teve o seu valor ajustado. O Quadro 15 apresenta o resultado da fase de ajuste de hiperparâmetros para cada série (apenas os parâmetros ajustados foram listados).

Uma vez que os hiperparâmetros foram ajustados, os modelos foram treinados com o conjunto de treinamento de cada série, obtendo-se assim 40 modelos (cada uma das 10 séries foram modeladas com os 4 algoritmos escolhidos). Vale ressaltar que para os modelos FAM e EAM foi utilizado apenas uma época de treinamento.

Quadro 14 – Definição do espaço de busca dos hiperparâmetros

Modelo	Hiperparâmetro	Descrição	Intervalo
FAM	α	Parâmetro de escolha	[0.001] (valor fixo)
FAM	ρ_a	Parâmetro de vigilância do módulo ART_a	[0] (valor fixo)
FAM	ρ_b	Parâmetro de vigilância do módulo ART_b	[0.99] (valor fixo)
FAM	β	Taxa de aprendizagem	$[0, 1]$ (distribuição uniforme)
EAM	ρ_a	Parâmetro de vigilância do módulo ART_a	[1] (valor fixo)
EAM	ρ_b	Parâmetro de vigilância do módulo ART_b	[0.01] (valor fixo)
EAM	β	Taxa de aprendizagem	$[0, 1]$ (distribuição uniforme)
KNN	k	Quantidade de vizinhos mais próximos	$[1, 15]$
SNAIVE	–	Modelo não possui hiperparâmetros	–

Fonte – autoria própria

Quadro 15 – Resultado da etapa de ajuste de hiperparâmetros

	FAM	**EAM**	**KNN**
	Hiperparâmetro		
	β	β	k
GLORIA_ALMOCO	0.0247	0.0877	2
PONTAL_ALMOCO	0.0013	0.0704	12
PONTAL_CAFE	0.7141	0.1264	2
PONTAL_JANTAR	0.0793	0.1642	4
STAMONICA_ALMOCO	0.9468	0.1222	2
STAMONICA_CAFE	0.5888	0.0300	3
STAMONICA_JANTAR	0.0015	0.2680	3
UMU_ALMOCO	0.1133	0.1883	3
UMU_CAFE	0.0517	0.1434	1
UMU_JANTAR	0.1989	0.0226	2

Fonte – autoria própria

5.5.2 Avaliação dos modelos

5.5.2.1 Análise de resíduos

A análise dos resíduos busca avaliar a qualidade de ajuste do modelo. Segundo Hyndman e Athanasopoulos (2018), duas propriedades devem ser satisfeitas:

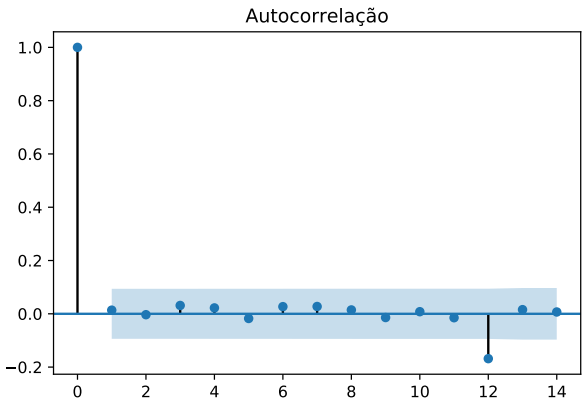
1. Os resíduos não são correlacionados. Se existe correlação no resíduo significa que informações importantes não foram capturadas pelos modelos, podendo comprometer a qualidade das previsões.

2. Os resíduos possuem média zero. A média residual diferente de zero aponta para um modelo que pode gerar previsões tendenciosas.

Para avaliar a primeira propriedade, foi utilizado o teste Ljung-Box (LJUNG; BOX, 1978) com atraso máximo de 14 dias e nível de significância a 5%.

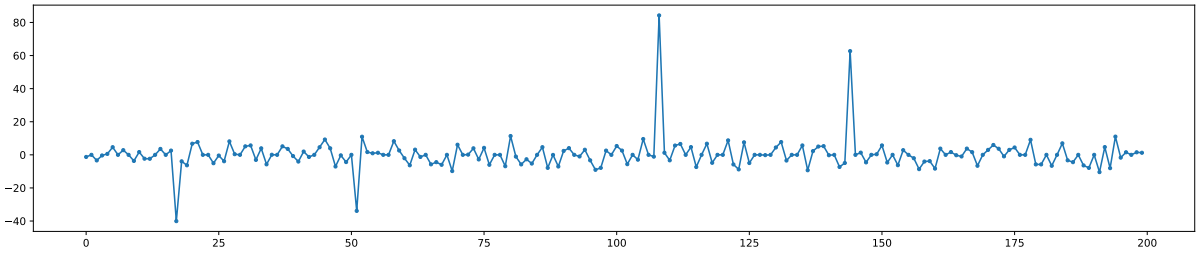
A título de exemplo, estão apresentados nos Gráficos 18 e 19 os resultados da atividade de análise de resíduos para a série `UMU_ALMOCO` modelada com a RNA FAM. Como pode ser observado, apesar de existir uma autocorrelação significativa no atraso 12, o teste de Ljung-Box resultou em um valor-p maior que o nível de significância escolhido. Neste caso aceita-se a hipótese nula, de que o resíduo não apresenta autocorrelação.

Gráfico 18 – Correlograma do resíduo do modelo FAM para a série `UMU_ALMOCO`. Resultados do teste de Ljung-Box : $Q = 14.72$, valor-p $= 0.3971$.



Fonte – autoria própria

Gráfico 19 – Resíduo do modelo FAM para a série `UMU_ALMOCO`. Valor da média : 0.1288



Fonte – autoria própria

5.5.2.2 Avaliação da acurácia das previsões

Ao contrário da análise de resíduos que utiliza os dados de treinamento para avaliar a qualidade de ajuste, a avaliação da acurácia de previsões utiliza o conjunto de teste (obtido na atividade de pré-processamento dos dados). Uma vez que estes dados não foram utilizados no ajuste dos modelos, esta avaliação pode fornecer uma estimativa da qualidade das previsões em situações reais de utilização (HYNDMAN; ATHANASOPOULOS, 2018).

Assumindo que neste ponto o modelo se encontra devidamente ajustado com o conjunto de treinamento, foi adotado o seguinte procedimento para a avaliação das previsões:

1. Na primeira iteração o modelo é utilizado para gerar previsões para os primeiros 30 dias do conjunto de teste. As previsões são comparadas (utilizando as métricas RMSE e MAE) com os valores reais do conjunto de teste.

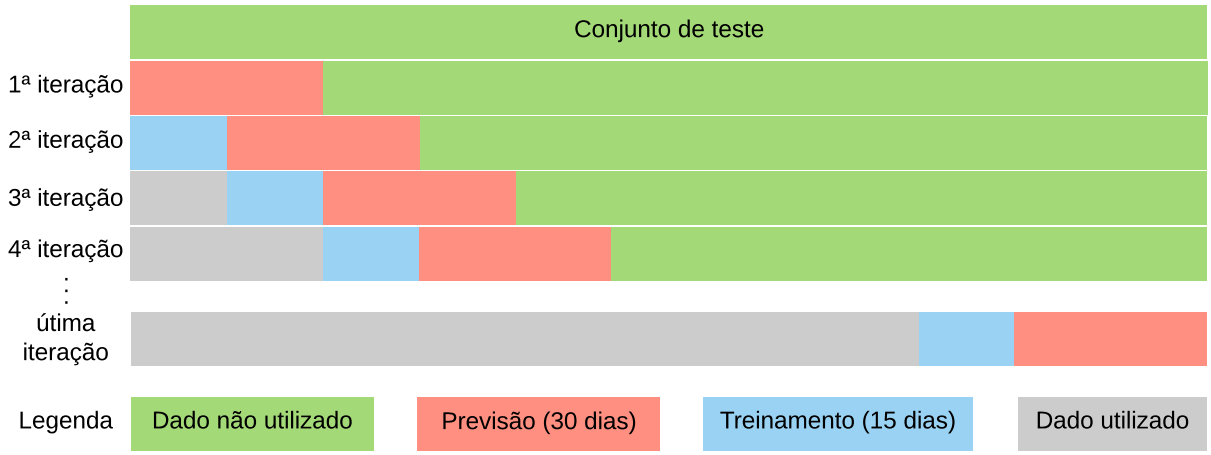
2. Na segunda iteração o modelo é atualizado com os primeiros 15 dias do conjunto de teste e então utilizado para gerar previsões para os próximos 30 dias. Do mesmo modo é feita a comparação, utilizando as mesmas métricas, com os dados reais do conjunto de teste.

3. O procedimento do item 2 se repete, iniciando-se pela última observação utilizada no treinamento, até que se utilize todo conjunto de teste.

4. O valor final de cada métrica é obtido pela média dos valores correspondentes calculados em cada iteração.

A Figura 29 ilustra de forma gráfica o procedimento descrito.

Figura 29 – Representação gráfica do procedimento utilizado para avaliação da acurácia das previsões com treinamento continuado.



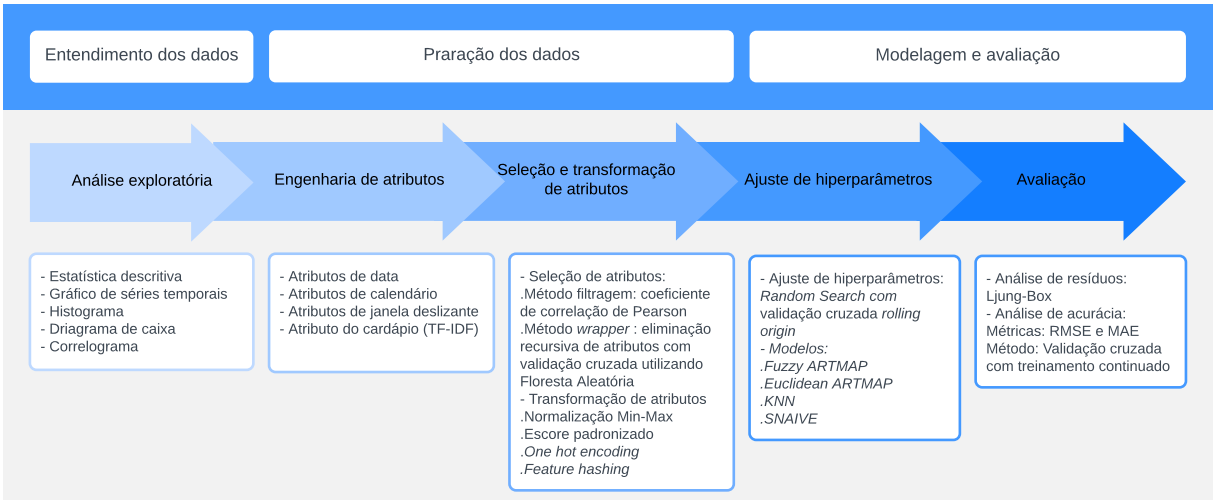
Fonte – autoria própria

Como pode ser constatado, as previsões geradas no procedimento de teste possuem horizonte de 30 dias. No entanto, os modelos foram treinados para prever apenas um passo adiante. Para lidar com esta limitação foi utilizada a estratégia de previsão recursiva de múltiplos passos. Nesta abordagem, a previsão realizada no passo $t-1$ é utilizada no passo t, sucessivamente, até se atingir o horizonte desejado (BONTEMPI; Ben Taieb; Le Borgne, 2013).

5.6 Visão geral

Com o objetivo de apresentar ao leitor uma visão geral da metodologia empregada na construção dos modelos de previsão, estão apresentadas na Figura 30 as principais fases e atividades do processo bem como as ferramentas utilizadas em cada uma delas.

Figura 30 – Visão geral do processo de elaboração dos modelos



Fonte – autoria própria

5.7 Discussão dos resultados

Os resultados obtidos na atividade de avaliação dos modelos (seção 5.5.2) estão listados na Tabela 3. Para cada série temporal do RU e para cada algoritmo utilizado na comparação, estão listados os valores da acurácia das previsões (métricas RMSE e MAE) bem como a análise de resíduos. Os modelos, dentro de cada agrupamento, estão ordenados de forma ascendente pela métrica RMSE. Visto que estas métricas avaliam o erro cometido pelos modelos comparando-os com dados reais, quanto menor o resultado, melhor é a avaliação.

A análise de resíduos foi realizada utilizando o teste de Ljung-Box a um nível de significância de 5% com atraso máximo de 14 dias. Na tabela de resultados a coluna **Q** representa o valor do teste estatístico.

Observou-se que em seis das dez séries, o algoritmo KNN apresentou melhores resultados quanto à métrica RMSE. No entanto, em apenas duas (`GLORIA_ALMOCO` e `STAMONICA_CAFE`) o mesmo obteve resultados satisfatórios quanto à análise de resíduos (valor-p $> 5\%$). Tal fato demonstra que o algoritmo, de forma geral, não atingiu a qualidade de ajuste suficiente, deixando de assimilar informações importantes das séries temporais.

Tabela 3 – Resultados da atividade de avaliação dos modelos

(continua)

		Acurácia de previsão		Análise de resíduos		
Série	**Modelo**	**RMSE**	**MAE**	**Q**	**valor-p**	**Média**
GLORIA_ALMOCO	KNN	12,2071	7,5400	17,4207	0,2344	-0,1749
	EAM	13,9151	9,0077	36,8839	0,0008	-0,2071
	FAM	14,0657	9,1333	19,9450	0,1319	-0,0000
	SNAIVE	16,5141	9,6133	179,0109	0,0000	16,1145
PONTAL_ALMOCO	KNN	70,4844	48,2851	47,6161	0,0000	-1,0789
	FAM	75,5916	52,4267	6,3344	0,9573	-1,9953
	EAM	91,9468	61,6404	1,5268	1,0000	-3,5558
	SNAIVE	126,9010	95,2939	899,5743	0,0000	-37,0069
PONTAL_CAFE	KNN	9,4398	6,1303	25,5538	0,0295	-0,0506
	SNAIVE	13,6719	9,6394	1731,3192	0,0000	-8,2675
	FAM	13,7168	9,7096	23,4451	0,0534	-0,1332
	EAM	14,1364	9,8212	165,4104	0,0000	-0,7710
PONTAL_JANTAR	KNN	39,5702	22,9697	24,5919	0,0388	0,5514
	FAM	59,9303	35,3026	20,9272	0,1035	-0,7109
	EAM	61,7316	40,9183	2,8055	0,9994	-0,8431
	SNAIVE	78,9037	53,5091	788,5659	0,0000	-16,8342
STAMONICA_ALMOCO	FAM	227,7325	136,8085	11,7835	0,6237	-0,1059
	KNN	236,5037	149,9833	38,2109	0,0005	3,7431
	EAM	335,7102	216,7698	22,1633	0,0753	-24,4382
	SNAIVE	367,4114	256,0182	527,1554	0,0000	-58,4954
STAMONICA_CAFE	KNN	13,9760	8,3636	15,9046	0,3192	0,2212
	FAM	14,2587	8,3274	4,2956	0,9934	0,0226
	EAM	16,5306	9,4199	19,4311	0,1491	-0,1579
	SNAIVE	19,8723	11,7121	1924,2376	0,0000	-2,7925
STAMONICA_JANTAR	KNN	114,7319	71,1030	35,9400	0,0011	1,5923
	FAM	126,5262	80,0684	3,1726	0,9987	-3,7956
	EAM	170,9846	107,9908	94,8509	0,0000	-13,3973
	SNAIVE	191,4557	113,8333	249,7706	0,0000	-70,4023
UMU_ALMOCO	FAM	117,7872	68,6516	14,7271	0,3971	0,1288
	KNN	126,7330	72,4626	66,4288	0,0000	-6,5627
	EAM	178,1053	109,2027	21,5601	0,0881	-14,3218
	SNAIVE	190,6711	135,5273	477,5998	0,0000	-21,0183
UMU_CAFE	FAM	9,8709	5,2846	19,7351	0,1387	-0,2998

Tabela 3 – Resultados da atividade de avaliação dos modelos

(conclusão)

		Acurácia de previsão		Análise de resíduos		
Série	**Modelo**	**RMSE**	**MAE**	**Q**	**valor-p**	**Média**
	SNAIVE	11,8964	6,8394	3474,4275	0,0000	-8,5123
	EAM	13,3669	8,2399	202,4384	0,0000	-1,3971
	KNN	18,1341	11,9667	0,4837	1,0000	0,2935
UMU_JANTAR	FAM	38,0886	22,4006	15,1888	0,3654	-0,6766
	EAM	48,1593	30,1744	6,5959	0,9492	-3,0508
	KNN	59,6184	36,9621	17,4836	0,2313	0,0386
	SNAIVE	63,8364	39,8030	321,7829	0,0000	-7,9541

Fonte – autoria própria

Por outro lado, o método KNN apresentou a melhor avaliação para a série GLORIA_ALMOCO que possui o menor número de observações (259). Por se tratar de um método baseado em instâncias (utiliza as observações do conjunto de treinamento sem nenhum tipo de processamento), o mesmo não possui parâmetros internos para serem ajustados, dispensando a fase de treinamento. Por esta razão, o método KNN pode apresentar melhor capacidade de generalização se comparado a outros métodos mais sofisticados, em situações onde o conjunto de treinamento é relativamente pequeno (ROKACH, 2010).

Por se tratar de um método extremamente simples o modelo SNAIVE obteve as piores avaliações (de forma geral), tanto no quesito acurácia quanto na análise de resíduos. No entanto, destaca-se a importância da utilização do método como linha de base e referência para comparação de desempenho em relação aos modelos mais sofisticados.

Os dados da Tabela 3 foram sumarizados na Tabela 4, obtida pela média aritmética de todas as métricas por modelo. Com isso foi possível avaliar o desempenho geral de cada algoritmo. Os dados sumarizados estão representados no gráfico de coordenadas paralelas da Figura 31 com o intuito de tornar mais evidente a diferença relativa entre os resultados obtidos por cada modelo. As linhas horizontais representam cada um dos modelos comparados, identificados pelas cores: preta (SNAIVE), azul (EAM), verde (FAM) e vermelha (KNN) conforme a legenda. As linhas verticais representam as métricas avaliadas: RMSE e MAE para a avaliação de acurácia e Q, "valor-p", "média resíduo" para avaliação da qualidade de ajuste. O ponto em que a linha horizontal cruza a linha vertical representa o valor da métrica, conforme a escala representada em cada linha vertical.

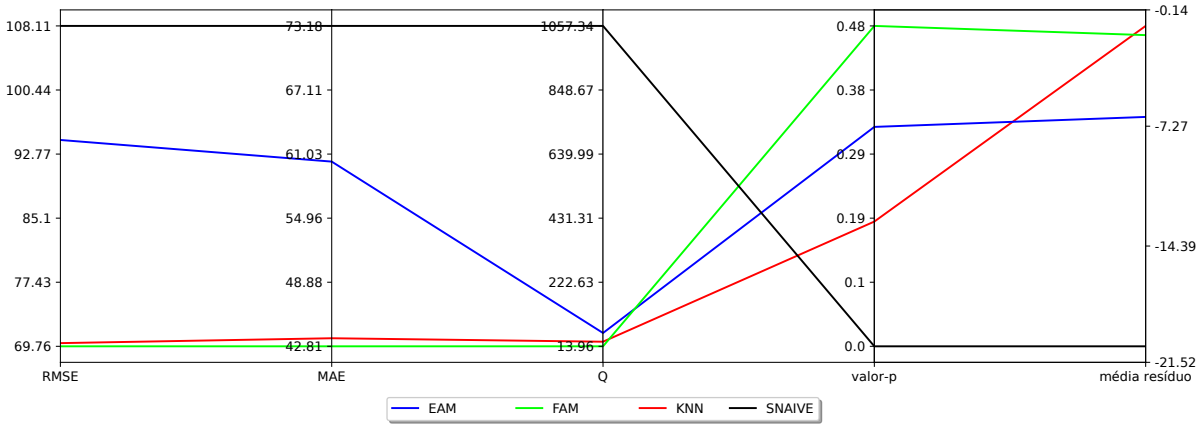
Observou-se que a rede neural FAM obteve resultados semelhantes ao algoritmo KNN quanto às métricas de acurácia. Apesar disso, pela análise dos resíduos, a rede FAM demonstrou melhor qualidade de ajuste (valores menores da estatística **Q**).

Tabela 4 – Desempenho geral dos modelos de previsão

	Acurácia de previsão		Análise de resíduos		
Modelo	**RMSE**	**MAE**	**Q**	**valor-p**	**Média**
FAM	69,7567	42,8113	13,9554	0,4763	-0,7565
KNN	70,1398	43,5766	28,9634	0,1854	-0,1427
EAM	94,4586	60,3185	57,3666	0,3261	-6,2140
SNAIVE	108,1134	73,1789	1057,3444	0,0000	-21,5169

Fonte – autoria própria

Figura 31 – Gráfico de coordenadas paralelas do desempenho geral de cada modelo.



Fonte – autoria própria

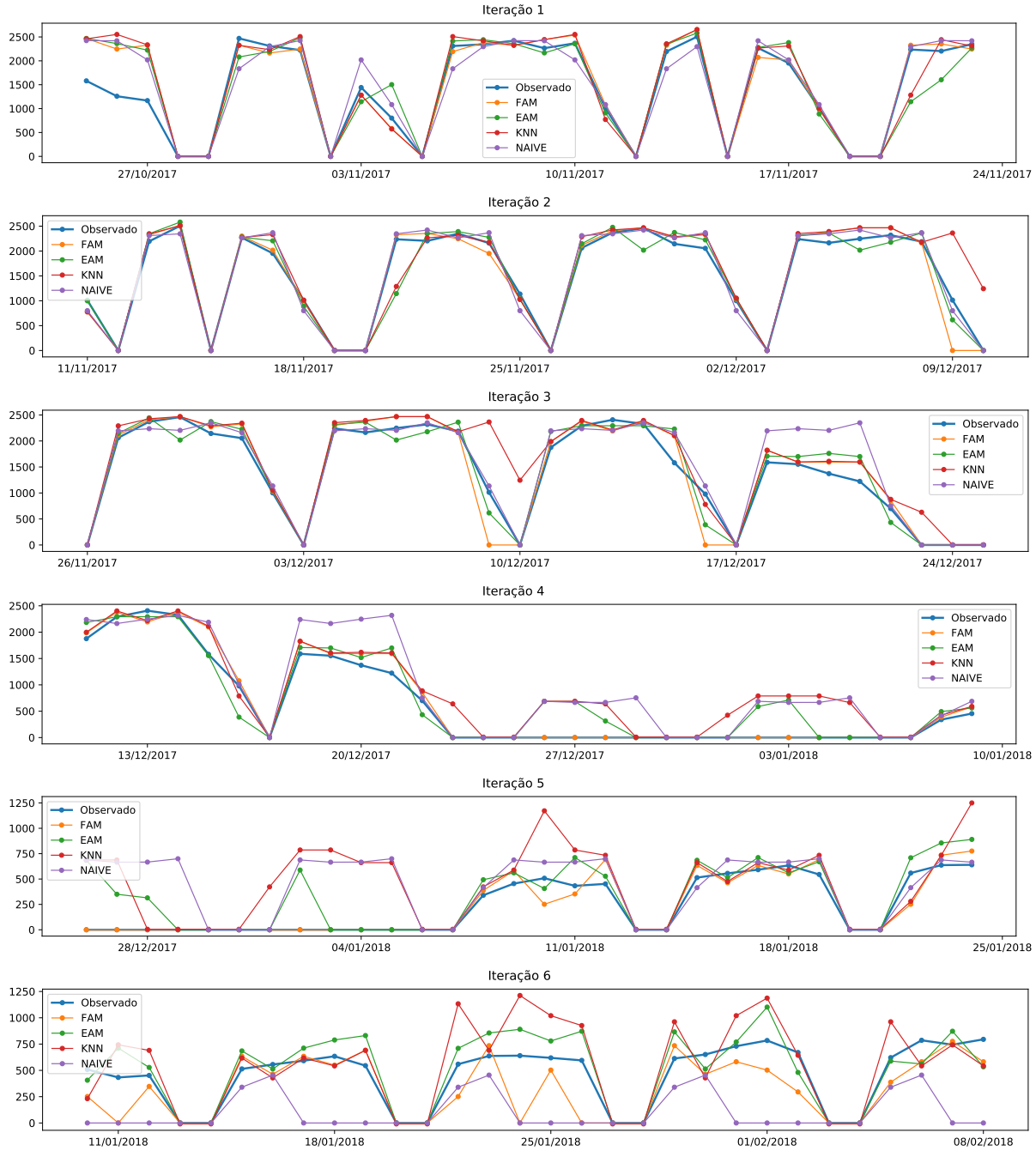
Apesar de que os algoritmos FAM e KNN obtiveram resultados estatisticamente semelhantes, a utilização da FAM é mais vantajosa. Como já colocado (seção 2.1), o aprendizado do algoritmo KNN é baseado em instâncias, o que implica no armazenamento de todo o conjunto de treinamento. Outra desvantagem do algoritmo KNN é que o mesmo precisa percorrer todo conjunto de dados para encontrar os K vizinhos mais próximos. Esta ação pode ser inviável dependendo do tamanho do conjunto de dados.

A rede neural EAM, apresentou desempenho inferior aos modelos FAM e KNN quanto às métricas de acurácia. No entanto, superou os métodos KNN e SNAIVE quanto à análise de resíduos.

A título de exemplo, as Figuras 32 e 33 ilustram graficamente o resultado dos testes dos algoritmos FAM, EAM, KNN e SNAIVE para a série temporal `STAMONICA_ALMOCO`, conforme procedimento descrito na subseção 5.5.2.2. A Figura 32 contém as seis primeiras iterações, enquanto que a Figura 33 as últimas cinco iterações. A linha azul simboliza a série reservada para o teste, portanto representa a quantidade de refeições que ocorreram de fato para todo o período. A linha laranja representa as previsões geradas para o modelo FAM, enquanto que as linhas verde, vermelha e roxa representam as previsões geradas pelos modelos EAM, KNN e SNAIVE respectivamente, conforme consta nas legendas em cada iteração.

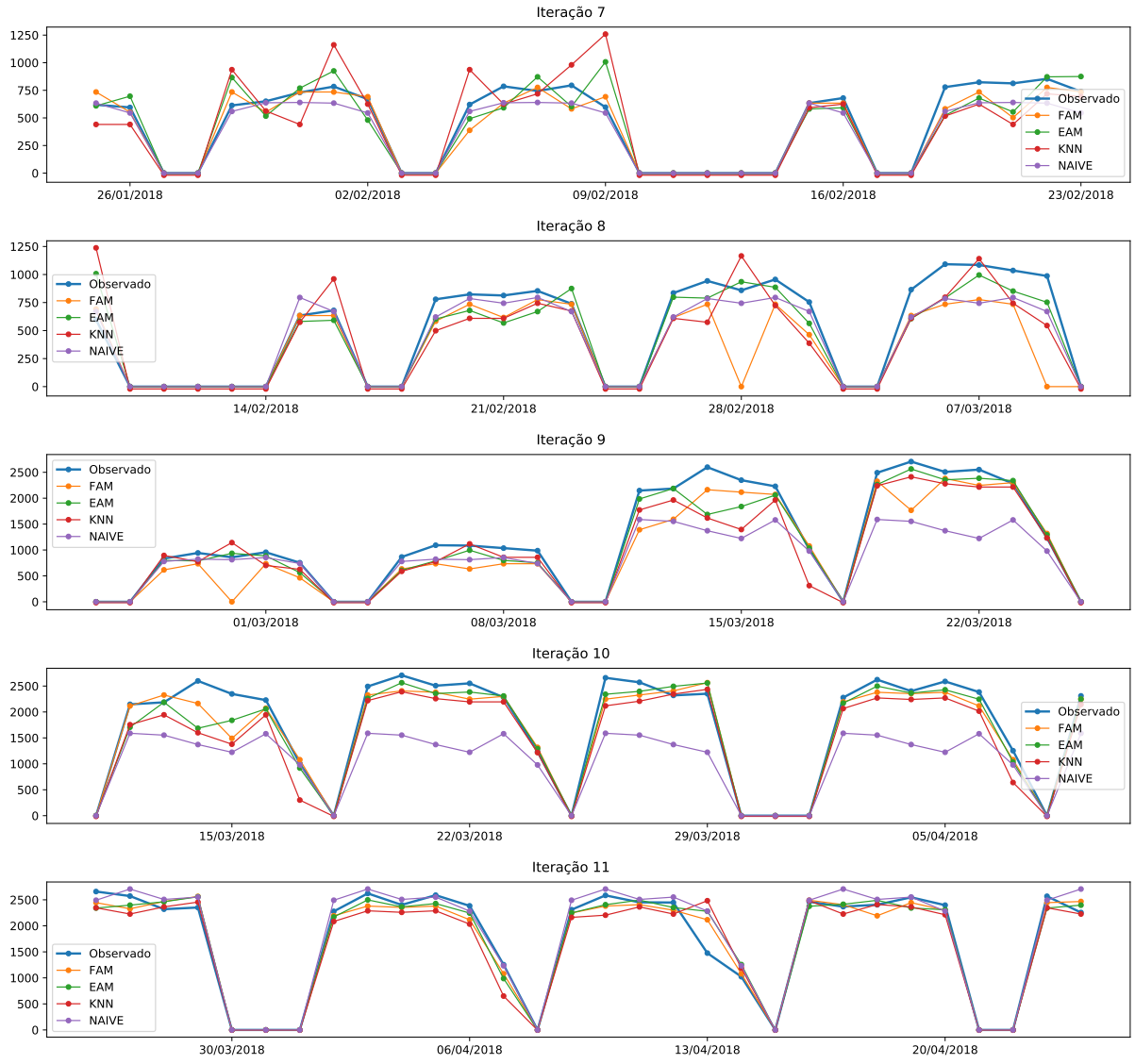
No exemplo ilustrado nas Figuras 32 e 33 a rede FAM obteve os melhores resultados (ver tabela 3). Observou-se que o modelo respondeu adequadamente às variações de demanda ao longo do período de testes (seis meses), notadamente nas semanas que antecedem ou sucedem o período de férias (22/12/2017 a 12/03/2018) onde ocorrem a redução e aumento gradativos da demanda de refeições. Todavia, durante o período de férias a rede neural respondeu de forma ruidosa se comparado aos períodos letivos. Tal comportamento pode ser atribuído possivelmente a um número de observações insuficientes que correspondam aos dias de férias no conjunto de treinamento, para que a rede possa generalizar corretamente para o período. Tal constatação indica para uma possível solução mista onde um modelo seria empregado para prever a demanda no período letivo e outro para o período de férias.

Figura 32 – Teste de acurácia dos algoritmos FAM, EAM, KNN e SNAIVE para a série `STAMONICA_ALMOCO`. Período de 25/10/2017 a 08/02/2018.



Fonte – autoria própria

Figura 33 – Teste de acurácia dos algoritmos FAM, EAM, KNN e SNAIVE para a série `STAMONICA_ALMOCO`. Período de 26/01/2018 a 24/08/2018.



Fonte – autoria própria

6 CONSIDERAÇÕES FINAIS

O Restaurante universitário (RU) da Universidade Federal de Uberlândia (UFU) desempenha um importante papel sócio-econômico oferecendo refeições de baixo custo à comunidade acadêmica da instituição. Como resultado, o RU contribui para a redução da evasão e retenção acadêmica, além de propiciar condições favoráveis e democráticas para as práticas das atividades de ensino, pesquisa e extensão. Mensalmente o RU fornece, em média, mais de 100.000 refeições entre café da manhã, almoço e jantar em quatro campi da universidade. Esta demanda é suprida através de contrato firmado com uma empresa terceirizada a um custo anual de R$ 13.369.487,52. A operacionalização do fornecimento das refeições exige um planejamento mensal por parte da administração do RU com o objetivo racionalizar a utilização dos recursos públicos. A previsão de demanda para o horizonte de 30 dias é uma importante informação para subsidiar os gestores na tomada de decisão.

Este trabalho utilizou técnicas de aprendizado de máquina para a elaboração de modelos de previsão de demanda diária de refeições do RU da UFU. Foram comparados os resultados de quatro algoritmos com capacidade de treinamento continuado: FAM, EAM, KNN e SNAIVE, dos quais, os dois últimos foram utilizados como linha de base. Os dados históricos da demanda diária foram divididos em dez séries temporais, baseado no campus e modalidade de refeição. Cada série foi analisada individualmente comparado-se os resultados obtidos por cada modelo.

A metodologia empregada na pesquisa foi baseada no processo CRISP-DM com adaptações específicas para a realidade de projetos com aprendizado de máquina. As principais atividades do processo envolvem a análise exploratória dos dados, pré-processamento (limpeza, engenharia, seleção e transformação de atributos), otimização dos hiperparâmetros e avaliação dos modelos. A avaliação dos modelos considerou dois critérios: a qualidade de ajuste e acurácia das previsões em regime de treinamento continuado.

Os resultados indicam para uma solução mista constituída pelos algoritmos FAM e KNN. O primeiro apresentou melhores resultados em oito das dez séries analisadas. Já o método KNN superou os demais em duas das dez séries. A análise das métricas dos modelos eleitos demonstraram que a metodologia empregada foi capaz de produzir modelos com boa qualidade de ajuste (resíduos não correlacionados e média residual próxima de

zero) além de uma capacidade de generalização adequada, aferida pelas métricas RMSE e MAE. Desta forma, podemos concluir que o objetivo geral deste trabalho foi alcançado satisfatoriamente.

Uma das limitações deste trabalho está em não elaborar intervalos de previsão ao longo do horizonte considerado. Esta informação fornece um intervalo de confiança dentro do qual se espera que a previsão esteja contida, dada uma probabilidade. Além do mais é importante salientar que o uso dos modelos está condicionado ao conhecimento prévio das variáveis explicativas para o horizonte de previsão desejado, tais como o cardápio e o calendário acadêmico.

Para o aprimoramento da pesquisa em trabalhos futuros, sugerimos a utilização de métodos baseados em comitês (*ensamble*) onde os resultados de um conjunto de modelos são agrupados com a finalidade de se aprimorar a capacidade de generalização. Como constatado na discussão dos resultados, outra possível melhoria pode ser alcançada modelando-se separadamente o período letivo e o período de férias. Adicionalmente, esta pesquisa poderá ser estendida para a elaboração de previsões de médio a longo prazo com objetivos estratégicos, como, por exemplo, a ampliação das unidades do RU ou a reavaliação contratual com a empresa terceirizada.

REFERÊNCIAS

ADHIKARI, R.; AGRAWAL, R. K. *An Introductory Study on Time Series Modeling and Forecasting*. [S.l.]: Lap Lambert Academic Publishing GmbH KG, 2013. 68 p. ISBN 9783659335082. 47, 71, 73, 75

ALBON, C. *Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning*. 1. ed. Sebastopol, CA: O'Reilly Media, 2018. ISBN 9781491989388. 98

ALBUQUERQUE, J. P. d. A. e.; FORTES, J. M. P.; FINAMORE, W. A. *Probabilidade, Variáveis Aleatórias e Processos Estocásticos*. 1. ed. Rio de Janeiro: PUC-Rio/Interciência, 2008. ISBN 978-85-7193-190-9. 73, 74

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, JMLR.org, v. 13, p. 281–305, 2012. ISSN 1532-4435. 55

BISGAARD, S.; KULAHCI, M. *Time Series Analysis and Forecasting by Example*. Hoboken: Wiley, 2011. 392 p. Disponível em: <https://doi.org/10.1002/9781118056943>. 70

BISHOP, C. M. *Pattern Recognition and Machine Learning*. 1. ed. Singapore: Springer, 2006. ISBN 9780387310732. 27, 28, 30, 47, 48, 49, 50, 53, 54, 97

BONTEMPI, G.; Ben Taieb, S.; Le Borgne, Y. A. Machine learning strategies for time series forecasting. In: *Lecture Notes in Business Information Processing*. Springer, Berlin, Heidelberg, 2013. v. 138 LNBIP, p. 62–77. ISBN 9783642363177. ISSN 18651348. Disponível em: <https://doi.org/10.1007/978-3-642-36318-4_3>. 110

BOX, G. E. P. et al. *Time series analysis : forecasting and control*. 5. ed. Hoboken: John Wiley & Sons, 2015. ISBN 9781118675021. 71

BRASIL. Constituição (1988). *Constituição da República Federativa do Brasil*. Brasília, DF: Senado, 1988. Disponível em: <http://www.planalto.gov.br/ccivil_03/Constituicao/ConstituicaoCompilado.htm>. Acesso em: 30 Mar. 2019. 22

BREIMAN, L. Random forests. *Machine Learning*, Kluwer Academic Publishers, v. 45, n. 1, p. 5–32, 2001. ISSN 08856125. 98

CARDOZO, J. E. M. Princípios constitucionais da administração pública. In: *Os 10 anos da Constituição federal*. [S.l.]: Editora Atlas, 1999. ISBN 9788522420780. 22

CARPENTER, G. A.; GROSSBERG, S. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, Academic Press, v. 37, n. 1, p. 54–115, jan 1987. ISSN 0734189X. Disponível em: <https://doi.org/10.1016/S0734-189X(87)80014-2>. 58

CARPENTER, G. A.; GROSSBERG, S. Art 2: Stable self-organization of pattern recognition codes for analog input patterns. *Applied optics*, v. 26, p. 4919–30, 12 1987. Disponível em: <https://doi.org/10.1364/AO.26.004919>. 59

CARPENTER, G. A.; GROSSBERG, S. Art 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, v. 3, p. 129–152, 12 1990. Disponível em: <https://doi.org/10.1016/0893-6080(90)90085-Y>. 59

CARPENTER, G. A.; GROSSBERG, S. The handbook of brain theory and neural networks. In: ARBIB, M. A. (Ed.). Cambridge, MA, USA: MIT Press, 1998. cap. Adaptive Resonance Theory (ART), p. 79–82. ISBN 0-262-51102-9. Disponível em: <http://dl.acm.org/citation.cfm?id=303568.303586>. 58

CARPENTER, G. A. et al. Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps. *IEEE Transactions on Neural Networks*, 1992. ISSN 19410093. Disponível em: <https://doi.org/10.1109/72.159059>. 59, 64, 65, 77

CARPENTER, G. A.; GROSSBERG, S.; REYNOLDS, J. H. Artmap: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, v. 4, p. 565–588, 1991. Disponível em: <https://doi.org/10.1016/0893-6080(91)90012-T>. 59

CARPENTER, G. A.; GROSSBERG, S.; ROSEN, D. B. Art 2-a: An adaptive resonance algorithm for rapid category learning and recognition. *Neural Networks*, v. 4, p. 493–504, 1991. Disponível em: <https://doi.org/10.1016/0893-6080(91)90045-7>. 59

CARPENTER, G. A.; GROSSBERG, S.; ROSEN, D. B. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, Pergamon, v. 4, n. 6, p. 759–771, jan 1991. ISSN 08936080. Disponível em: <https://doi.org/10.1016/0893-6080(91)90056-B>. 102

CHAMBERS, J. M. et al. *Graphical methods for data analysis*. Boca Raton: CRC Press, 2018. ISBN 9781315893204. Disponível em: <https://doi.org/10.1201/9781351072304>. 87, 89

CHAPELLE BERNHARD SCHöLKOPF, A. Z. O. *Semi-Supervised Learning*. 1. ed. Cambridge, Massachusetts: MIT Press, 2006. (Adaptive computation and machine learning). ISBN 978-0-262-03358-9. Disponível em: <https://doi.org/10.7551/mitpress/9780262033589.001.0001>. 29

CHAPMAN, P. et al. *CRISP-DM 1.0 Step-by-step data mining guide*. [S.l.], 2000. Disponível em: <http://www.crisp-dm.org/CRISPWP-0800.pdf>. 31, 32, 33, 77, 78

CHASE, C. *Demand-driven forecasting : a structured approach to forecasting*. [S.l.]: John Wiley & Sons, 2009. ISBN 9780470415023. 18, 19

CHATFIELD, C. *The Analysis Of Time Series - An Introduction*. 6. ed. Boca Raton: Chapman and Hall/CRC, 2003. (Chapman & Hall/CRC Texts in Statistical Science). ISBN 9780203491683. 70, 71, 74, 75, 84

CHIUSO, A.; PILLONETTO, G. System Identification: A Machine Learning Perspective. *Annual Review of Control, Robotics, and Autonomous Systems*, v. 2, n. 1, p. annurev–control–053018–023744, may 2019. ISSN 2573-5144. Disponível em: <https://doi.org/10.1146/annurev-control-053018-023744>. 28

CHRIST, M. et al. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). *Neurocomputing*, Elsevier, v. 307, p. 72–77, sep 2018. ISSN 0925-2312. Disponível em: <https://doi.org/10.1016/j.neucom.2018.03.067>. 96

CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, v. 2, n. 4, p. 303–314, 1989. ISSN 1435-568X. 19

DAVID, V. K.; RAJASEKARAN, S. *Pattern Recognition using Neural and Functional Networks*. 1. ed. Springer Berlin Heidelberg, 2009. (Studies in Computational Intelligence 160). ISBN 9783540851295. Disponível em: <https://doi.org/10.1007/978-3-540-85130-1>. 58

DOMINGOS, P. A few useful things to know about machine learning. *Communications of the ACM*, 2012. ISSN 00010782. Disponível em: <https://doi.org/10.1145/2347736.2347755>. 36

GROSSBERG, S. Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. *Biological Cybernetics*, v. 23, n. 3, p. 121–134, 1976. ISSN 1432-0770. Disponível em: <https://doi.org/10.1007/BF00344744>. 56

GROSSBERG, S. Adaptive pattern classification and universal recoding: II. Feedback, expectation, olfaction, illusions. *Biological Cybernetics*, v. 23, n. 4, p. 187–202, 1976. ISSN 1432-0770. Disponível em: <https://doi.org/10.1007/BF00340335>. 56

GROSSBERG, S. How Does a Brain Build a Cognitive Code? In: *Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition, and Motor Control*. Dordrecht: Springer Netherlands, 1982. p. 1–52. ISBN 978-94-009-7758-7. Disponível em: <https://doi.org/10.1007/978-94-009-7758-7_1>. 56

GROSSBERG, S. Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world. *Neural Netw.*, Elsevier Science Ltd., Oxford, UK, UK, v. 37, p. 1–47, jan. 2013. ISSN 0893-6080. Disponível em: <https://doi.org/10.1016/j.neunet.2012.09.017>. 21, 56

HASTIE ROBERT TIBSHIRANI, J. F. T. *The elements of statistical learning: Data mining, inference, and prediction*. 2. ed. [S.l.]: Springer, 2009. (Springer Series in Statistics). ISBN 9780387848587. 51

HILFIGER, J. J. *Graphing Data with R*. 1. ed. Sebastopol, CA: O'Reilly Media, 2015. ISBN 9781491922613. 87

HOENS, T. R.; POLIKAR, R.; CHAWLA, N. V. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, v. 1, n. 1, p. 89–101, 2012. ISSN 2192-6360. Disponível em: <https://doi.org/10.1007/s13748-011-0008-0>. 21

HYNDMAN, R.; ATHANASOPOULOS, G. *Forecasting: principles and practice*. 2. ed. [S.l.]: OTexts, 2018. ISBN 9780987507112. 19, 47, 77, 91, 106, 108, 109

JAMES, G. et al. *An Introduction to Statistical Learning with Applications in R*. [S.l.]: Springer, 2017. ISBN 9781461471370. 100

KAUFMAN, S. et al. Leakage in data mining. *ACM Transactions on Knowledge Discovery from Data*, ACM Press, New York, New York, USA, v. 6, n. 4, p. 1–21, 2012. ISSN 15564681. Disponível em: <https://doi.org/10.1145/2382577.2382579>. 94

KENAYA, R.; CHEOK, K. C. Euclidean art neural networks. In: CITESEER. *Proceedings of the World Congress on Engineering and Computer Science*. [S.l.], 2008. p. 963–968. 62, 64, 102

KENAYA, R.; CHEOK, K. C. Euclidean artmap based target tracking control system. In: *2008 IEEE International Conference on Electro/Information Technology*. [s.n.], 2008. p. 41–47. ISSN 2154-0357. Disponível em: <https://doi.org/10.1109/EIT.2008.4554265>. 68, 77

KLINE, D. M. Methods for Multi-Step Time Series Forecasting Neural Networks. In: *Neural Networks in Business Forecasting*. Hershey, PA: IGI Global, 2003. p. 226–250. ISBN 9781591401766. Disponível em: <https://doi.org/10.4018/978-1-59140-176-6.ch012>. 79

KOLASSA, S.; SIEMSEN, E. *Demand Forecasting for Managers*. [S.l.]: Business Expert Press, 2016. 200 p. ISBN 978-1-606-49502-5. 18, 19

LECUN, Y.; CORTES, C. MNIST handwritten digit database. 2010. Disponível em: <http://yann.lecun.com/exdb/mnist/>. 26

LJUNG, G. M.; BOX, G. E. P. On a measure of lack of fit in time series models. *Biometrika*, v. 65, n. 2, p. 297–303, 08 1978. ISSN 0006-3444. Disponível em: <https://doi.org/10.1093/biomet/65.2.297>. 108

MAINDONALD, J.; BRAUN, W. J. *Data Analysis and Graphics Using R: An Example-Based Approach*. 3. ed. [S.l.]: Cambridge University Press, 2010. (Cambridge Series in Statistical and Probabilistic Mathematics). ISBN 9780521762939. 97

MANNING, C. D.; RAGHAVAN, P.; SCHUTZE, H. *Introduction to information retrieval*. 1. ed. Cambridge University Press, 2008. ISBN 9780521865715. Disponível em: <https://doi.org/10.1017/CBO9780511809071>. 45

MINEAULT, P. *Fast 1D and 2D data binning in Matlab – xcorr: comp neuro*. 2013. Disponível em: <https://xcorr.net/2013/12/23/fast-1d-and-2d-data-binning-in-matlab/>. Acesso em: 01 jan. 2019. 39

MITCHELL, T. M. *Machine Learning*. 1. ed. New York, NY: McGraw-Hill, 1997. (McGraw-Hill series in computer science). ISBN 0070428077. 25, 27

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. *Foundations of Machine Learning*. 1. ed. Cambridge: The MIT Press, 2012. ISBN 9780262018258. 25, 29

MOORE, B. Art 1 and pattern clustering. In: *Proceedings of the 1988 Connectionist Models Summer School*. [S.l.: s.n.], 1989. p. 174–185. 61

MORENO, A. L. U. *Análise da estabilidade transitória via rede neural Art-Artmap fuzzy Euclidiana modificada com treinamento continuado*. 143 f. p. Tese (Doutorado), 2010. Disponível em: <http://hdl.handle.net/11449/100305>. 62

MORETTIN, P. A.; de Castro Toloi, C. M. *Análise de séries temporais*. 2. ed. São Paulo: Edgard Blucher, 2006. (ABE - Projeto Fisher). ISBN 9788521203896. 74

MUKAKA, M. M. Statistics corner: A guide to appropriate use of correlation coefficient in medical research. *Malawi Medical Journal*, Medical Association of Malawi, v. 24, n. 3, p. 69–71, sep 2012. ISSN 19957262. 45

NIST. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (U.S.), AND INTERNATIONAL SEMATECH. *Engineering Statistics Handbook*. Gaithersburg, Md., 2003. Disponível em: <http://www.itl.nist.gov/div898/handbook/>. Acesso em: 12 abr. 2018. 87

PAL, A.; PRAKASH, P. *Practical Time Series Analysis: Master Time Series Data Processing, Visualization, and Modeling using Python*. Birmingham: Packt Publishing, 2017. ISBN 9781788294195. 84

RAJASEKARAN, S.; PAI, G. V. *NEURAL NETWORKS, FUZZY SYSTEMS AND EVOLUTIONARY ALGORITHMS : Synthesis and Applications*. 2. ed. [S.l.]: PHI Learning, 2017. ISBN 9788120353343. 57, 60

RASCHKA, S.; MIRJALILI, V. *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow*. 2. ed. Birmingham: Packt Publishing, 2017. 622 p. ISBN 978-1-78712-593-3. 77

REID, R. D.; SANDERS, N. R. *Operations Management: An Integrated Approach*. 5. ed. [S.l.]: John Wiley & Sons, 2015. ISBN 9781118952610. 20

ROKACH, L. *Pattern classification using ensemble methods*. 1. ed. Singapore: WS, 2010. (Series in Machine Perception and Artifical Intelligence). ISBN 9789814271066. Disponível em: <https://doi.org/10.1142/7238>. 113

SARKAR, D. *Text Analytics with Python: A Practical Real-World Approach to Gaining Actionable Insights from your Data*. 1. ed. Bangalore, Karnataka: Apress, 2016. ISBN 9781484223871. Disponível em: <https://doi.org/10.1007/978-1-4842-2388-8>. 97

SARKAR, D.; BALI, R.; SHARMA, T. *Practical Machine Learning with Python: A Problem-Solver's Guide to Building Real-World Intelligent Systems*. 1st. ed. Berkely, CA, USA: Apress, 2017. ISBN 9781484232064. Disponível em: <https://doi.org/10.1007/978-1-4842-3207-1_1>. 25, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 41, 42, 43, 45, 46, 51, 52, 54, 55, 77, 78, 94, 97, 100, 101, 106

SCOPUS. 2019. Disponível em: <http://www.scopus.com/>. Acesso em: 24 mar. 2019. 20

SERRANO-GOTARREDONA, T.; LINARES-BARRANCO, B.; ANDREOU, A. G. *Adaptive Resonance Theory Microchips : Circuit Design Techniques*. [S.l.]: Springer US, 1998. 234 p. ISBN 9781461346722. 62, 67

SHUMWAY, R. H.; STOFFER, D. S. *Time Series Analysis and Its Applications*. 4. ed. Cham: Springer International Publishing, 2017. (Springer Texts in Statistics). ISSN 01621459. ISBN 978-3-319-52451-1. Disponível em: <https://doi.org/10.1007/978-3-319-52452-8>. 75, 76

SIMON, P. *Too Big to Ignore: The Business Case for Big Data*. 1. ed. Hoboken, New Jersey: Wiley, 2013. ISBN 9781118638170. Disponível em: <https://doi.org/10.1002/9781119204039>. 25

STAńCZYK, L. C. J. e. U. *Feature Selection for Data and Pattern Recognition*. 1. ed. [S.l.]: Springer-Verlag Berlin Heidelberg, 2015. (Studies in Computational Intelligence 584). ISBN 978-3-662-45620-0. 45, 46

SWAMYNATHAN, M. *Mastering Machine Learning with Python in six Steps*. 1. ed. Bangalore, Karnataka: Apress, 2017. ISBN 978-1-4842-2866-1. Disponível em: <https://doi.org/10.1007/978-1-4842-2866-1>. 25

TSAI, C.; WANG, S. Stock price forecasting by hybrid machine learning techniques. In: *Proceedings of the International MultiConference of Engineers and Computer Scientists*. [S.l.: s.n.], 2009. v. 1, n. 755, p. 60. 28

TUKEY, J. W. *Exploratory Data Analysis*. 1. ed. [S.l.]: Addison Wesley, 1977. ISBN 0201076160. 34

UNIVERSIDADE FEDERAL DE UBERLÂNDIA. Compras, Licitações e Contratos. *Contrato 033/15*. Uberlândia, MG, 2018. Disponível em: <http://www.licitacoes.ufu.br/node/738>. Acesso em: 14 Ago. 2018. 22

UNIVERSIDADE FEDERAL DE UBERLÂNDIA. Pró-Reitoria de Assistência Estudantil. Restaurantes Universitários. *Cardápio*. Uberlândia, MG, 2018. Disponível em: <http://www.proae.ufu.br/ru/cardapios>. Acesso em: 5 set. 2018. 83

UNIVERSIDADE FEDERAL DE UBERLÂNDIA. Pró-reitoria de Graduação. *Calendário Acadêmico*. Uberlândia, MG, 2018. Disponível em: <http://www.prograd.ufu.br/central-de-conteudos/documentos/calendario>. Acesso em: 20 mai. 2018. 81

WIEDERHOLD, G.; MCCARTHY, J. Arthur Samuel: Pioneer in Machine Learning. *IBM Journal of Research and Development*, v. 36, n. 3, p. 329–331, 1992. ISSN 0018-8646. Disponível em: <https://doi.org/10.1147/rd.363.0329>. 25

Wikipedia contributors. *Data transformation (statistics) — Wikipedia, The Free Encyclopedia*. 2018. [Online; accessed 31-January-2019]. Disponível em: <https://en.wikipedia.org/w/index.php?title=Data_transformation_(statistics)&oldid=873670993>. 40

ZHANG, G. P. Business Forecasting with Artificial Neural Networks: An Overview. In: *Neural Networks in Business Forecasting*. Hershey, PA: IGI Global, 2004. p. 1–22. ISBN 9781591401766. 21

ZHENG, A.; CASARI, A. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. 1. ed. Sebastopol, CA: O'Reilly Media, 2018. ISBN 9781491953242. 36, 38, 40, 41, 42, 43, 44

APÊNDICE A – CONSULTA SQL UTILIZADA NA CRIAÇÃO DOS ATRIBUTOS DE CALENDÁRIO

Código 2 – Consulta SQL submetida para a criação dos atributos de calendário

```
WITH
    feature_pos_feriado_prolongado AS
    (
        SELECT
            e.fim + 1 AS data,
            0 AS feature_feriado,
            0 AS feature_ferias,
            0 AS feature_greve,
            0 AS feature_segunda_vespera_feriado,
            0 AS feature_sexta_pos_feriado,
            0 AS feature_vespera_ferias,
            0 AS feature_pos_ferias,
            0 AS feature_vespera_feriado_prolongado,
            1 AS feature_pos_feriado_prolongado,
            cidade_id
        FROM
            evento e
        WHERE
            evento_tipo_id = 1
        AND (extract(dow FROM fim) IN (5,6) OR extract(dow FROM inicio) = 1)
    )
    ,
    feature_vespera_feriado_prolongado AS
    (
        SELECT
            generate_series( e.inicio - 2, e.inicio - 1, '1 day':: interval):: DATE AS data,
            0 AS feature_feriado,
            0 AS feature_ferias,
            0 AS feature_greve,
            0 AS feature_segunda_vespera_feriado,
            0 AS feature_sexta_pos_feriado,
            0 AS feature_vespera_ferias,
            0 AS feature_pos_ferias,
            1 AS feature_vespera_feriado_prolongado,
            0 AS feature_pos_feriado_prolongado,
            cidade_id
        FROM
            evento e
        WHERE
            evento_tipo_id = 1
```

```
    AND extract(dow FROM fim) IN (5)
)
,
feature_pos_ferias AS
(
    SELECT
        generate_series( e.fim + 1, e.fim + 7, '1 day':: interval):: DATE AS data,
        0 AS feature_feriado,
        0 AS feature_ferias,
        0 AS feature_greve,
        0 AS feature_segunda_vespera_feriado,
        0 AS feature_sexta_pos_feriado,
        0 AS feature_vespera_ferias,
        1 AS feature_pos_ferias,
        0 AS feature_vespera_feriado_prolongado,
        0 AS feature_pos_feriado_prolongado,
        cidade_id
    FROM
        evento e
    WHERE
        evento_tipo_id = 2
)
,
feature_vespera_ferias AS
(
    SELECT
        generate_series( e.inicio - 13, e.inicio - 1, '1 day':: interval):: DATE AS data,
        0 AS feature_feriado,
        0 AS feature_ferias,
        0 AS feature_greve,
        0 AS feature_segunda_vespera_feriado,
        0 AS feature_sexta_pos_feriado,
        CAST(extract(doy FROM e.inicio) - extract(doy FROM generate_series( e.inicio - 13,
        e.inicio - 1, '1 day':: interval):: DATE) AS INTEGER) / 7 + 1 AS
            feature_vespera_ferias,
        0 AS feature_pos_ferias,
        0 AS feature_vespera_feriado_prolongado,
        0 AS feature_pos_feriado_prolongado,
        cidade_id
    FROM
        evento e
    WHERE
        evento_tipo_id = 2
)
,
feature_segunda_vespera_feriado AS
```

```
(
    SELECT
        inicio-1 AS data,
        0 AS feature_feriado,
        0 AS feature_ferias,
        0 AS feature_greve,
        1 AS feature_segunda_vespera_feriado,
        0 AS feature_sexta_pos_feriado,
        0 AS feature_vespera_ferias,
        0 AS feature_pos_ferias,
        0 AS feature_vespera_feriado_prolongado,
        0 AS feature_pos_feriado_prolongado,
        cidade_id
    FROM
        evento e
    WHERE
        evento_tipo_id = 1
    AND extract(dow FROM inicio) = 2
)
,
feature_sexta_pos_feriado AS
(
    SELECT
        fim+1 AS data,
        0 AS feature_feriado,
        0 AS feature_ferias,
        0 AS feature_greve,
        0 AS feature_segunda_vespera_feriado,
        1 AS feature_sexta_pos_feriado,
        0 AS feature_vespera_ferias,
        0 AS feature_pos_ferias,
        0 AS feature_vespera_feriado_prolongado,
        0 AS feature_pos_feriado_prolongado,
        cidade_id
    FROM
        evento e
    WHERE
        evento_tipo_id = 1
    AND extract(dow FROM fim)=4
)
,
features_feriado_ferias_greve AS
(
    SELECT DISTINCT
        generate_series( e.inicio, e.fim, '1 day':: interval):: DATE AS data,
        CASE evento_tipo_id
```

```
            WHEN 1
            THEN 1
            ELSE 0
        END AS feature_feriado,
        CASE evento_tipo_id
            WHEN 2
            THEN 1
            ELSE 0
        END AS feature_ferias,
        CASE evento_tipo_id
            WHEN 3
            THEN 1
            ELSE 0
        END AS feature_greve,
        0 AS feature_segunda_vespera_feriado,
        0 AS feature_sexta_pos_feriado,
        0 AS feature_vespera_ferias,
        0 AS feature_pos_ferias,
        0 AS feature_vespera_feriado_prolongado,
        0 AS feature_pos_feriado_prolongado,
        cidade_id
    FROM
        evento e
    ORDER BY
        data
)
,
features_union AS
(
    SELECT
        *
    FROM
        feature_pos_feriado_prolongado
    UNION ALL
    SELECT
        *
    FROM
        feature_segunda_vespera_feriado
    UNION ALL
    SELECT
        *
    FROM
        feature_sexta_pos_feriado
    UNION ALL
    SELECT
        *
```

```
    FROM
        features_feriado_ferias_greve
    UNION ALL
    SELECT
        *
    FROM
        feature_pos_ferias
    UNION ALL
    SELECT
        *
    FROM
        feature_vespera_ferias
    UNION ALL
    SELECT
        *
    FROM
        feature_vespera_feriado_prolongado
    ORDER BY
        data
)
,
features AS
(
    SELECT
        data,
        MAX(feature_feriado)                AS feature_feriado,
        MAX(feature_ferias)                 AS feature_ferias,
        MAX(feature_greve)                  AS feature_greve,
        MAX(feature_segunda_vespera_feriado) AS feature_segunda_vespera_feriado,
        MAX(feature_sexta_pos_feriado)      AS feature_sexta_pos_feriado,
        MAX(feature_vespera_ferias)         AS feature_vespera_ferias,
        MAX(feature_pos_ferias)             AS feature_pos_ferias,
        MAX(feature_vespera_feriado_prolongado) AS feature_vespera_feriado_prolongado,
        MAX(feature_pos_feriado_prolongado) AS feature_pos_feriado_prolongado,
        cc.cidade_id
    FROM
        features_union fu
    INNER JOIN
        cidade_cidade cc
    ON
        fu.cidade_id = cc.grupo_cidade_id
    GROUP BY
        data,
        cc.cidade_id
    ORDER BY
        data
```

```
    )
SELECT DISTINCT
    data,
    feature_feriado,
    feature_ferias,
    feature_greve,
    feature_vespera_ferias AS feature_semana_vespera_ferias,
    feature_pos_ferias   AS feature_semana_pos_ferias,
    feature_vespera_feriado_prolongado,
    feature_segunda_vespera_feriado,
    feature_sexta_pos_feriado,
    cidade_id
FROM
    features
ORDER BY
    data
```