Martim Afonso Gouveia Sousa

Previsão de Vendas de Peças de Substituição para Otimização da Gestão de Stock

Forecasting of Spare Parts Components for Storage Optimization

Universidade de Aveiro Departamento de Matemática
2021

Martim Afonso Gouveia Sousa

Previsão de Vendas de Peças de Substituição para Otimização da Gestão de Stock

Forecasting of Spare Parts Components for Storage Optimization

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Matemática e Aplicações, realizada sob a orientação científica do Doutor Eugénio Alexandre Miguel Rocha, Professor Associado do Departamento de Matemática da Universidade de Aveiro, e da Eng.ª Maria João Lopes (supervisora), Engenheira de Software na Bosch.

o júri / the jury

presidente / president

Prof. Doutor Vasile Staicu

Professor Catedrático do Departamento de Matemática da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Eugénio Alexandre Miguel Rocha (orientador)

Professor Associado do Departamento de Matemática da Universidade de Aveiro

Prof. Doutor Fernão Rodrigues Vístulo de Abreu (arguente)

Professor Auxiliar do Departamento de Física da Universidade de Aveiro

agradecimentos / acknowledgements

Em primeira instância, agradeço ao Prof. Doutor Eugénio Rocha pela orientação, disponibilidade, ajuda técnica na fase de ETL e por me ter dado autonomia na tomada de decisões ao longo do projeto. O Prof. Doutor Eugénio Rocha é, inequivocamente, uma mais valia para o corpo docente do Departamento de Matemática, visto que acrescenta multidisciplinaridade à matemática, tornando-a mais abrangente e interessante.

Agradeço ao Eng.º Nelson Ferreira, à Eng.ª Maria João Lopes, ao Eng.º Paulo Baptista e restantes membros da Bosch pela simpatia com que me receberam nas reuniões, motivando-me sempre para a melhoria constante do projeto, e pela prontidão com que foram esclarecendo dúvidas pontuais.

Ao corpo docente do Departamento de Matemática que me acompanhou durante estes anos, um enorme obrigado pelo conhecimento partilhado, potenciando sempre a capacidade de abstração e raciocínio lógico. Saliento também a preocupação constante na atualização dos planos curriculares, de forma a acompanhar os novos desafios da matemática e as necessidades do mercado de trabalho.

Por último, mas não menos importante, agradeço aos meus pais e restante família pelo altruísmo e apoio dado em todos os momentos deste percurso.

Palavras Chave

Previsão do número de vendas de peças de substituição; Aprendizagem profunda em séries temporais; Aprendizagem de máquina em séries temporais; Previsão em séries temporais; Holt-Winters; FBProphet; Support vector machines; Multilayer perceptron; Convolutional neural network; XGBoost; Long short-term memory

Resumo

Este projeto tem como intuito prever o número de vendas de peças de substituição, associadas a equipamentos de termotecnologia, das próximas três semanas para a empresa Bosch Termotecnologia-Aveiro. Desta forma, a Bosch pode otimizar o planeamento de produção e stocks garantindo a satisfação do cliente final.

A nossa abordagem ao problema consiste no estudo das séries temporais de vendas destas peças de substituição, construindo modelos matemáticos capazes de captar os padrões existentes nos dados. Os modelos utilizados cobrem um largo espetro de metodologias, por forma a acomodar diferentes tipos de séries temporais. Concretamente, utilizámos metodologias clássicas como o Autoregressive integrated moving average (ARIMA) e Holt-Winters, mas também métodos de aprendizagem supervisionada como o Support vector regression (SVR) e Extreme gradient boosting (XGBoost) e ainda modelos de redes neuronais, nomeadamente Multilayer perceptron (MLP), Convolutional neural network (CNN) e Long short-term memory (LSTM).

Keywords

Spare parts demand forecasting; Deep learning for time series forecasting; Machine learning for time series forecasting; Time series forecasting; Holt-Winters; FBProphet; Support vector machines; Multilayer perceptron; Convolutional neural network; XGBoost; Long short-term memory

Abstract

This project aims to forecast the number of sales of spare parts, linked to thermotechnology equipment, in the next three weeks for the company Bosch Thermotechnology-Aveiro. In this way, Bosch can optimize production planning, reducing production and storage costs.

Our approach to the problem consists of studying the sales time series of these spare parts, in order to build mathematical models capable of capturing the patterns underlying in the data. The models used cover a wide spectrum of methodologies, with the purpose of fitting different types of time series. Specifically, we used classical methodologies such as ARIMA and Holt-Winters, but also supervised learning methods such as SVR and XGBoost and also deep learning approaches, namely MLP, CNN and LSTM.

Conteúdo

Lista de Acrónimos

AIC Akaike's information criterion. 25, 69

API Application programming interface. 73

ARIMA Autoregressive integrated moving average. 1, 6, 13, 14, 29, 31, 36, 54, 63, 69, 73, 77

BIC Bayesian information criterion. 25, 69

CNN Convolutional neural network. 1, 7, 58–61, 63, 69, 71–73

DFT Discrete Fourier transform. 21

EMV Estimação de máxima verosimilhança. 34

ETL Extract, transform, load. 1, 67

FAC Função de autocorrelação. 6, 7, 26, 27, 33, 36, 37

FACP Função de autocorrelação parcial. 6, 7, 26, 27, 36, 37, 69

FFT Fast Fourier transform. 21

KKT Karush-Kuhn-Tucker. 46, 48

KPSS Kwiatkowski–Phillips–Schmidt–Shin. 24, 69

LSTM Long short-term memory. 1, 7, 8, 61–63, 69, 71–73, 75, 76

MAPE Mean absolute percentage error. 64

MLP Multilayer perceptron. 1, 7, 14, 49, 50, 52–54, 57–61, 63, 64, 69, 71–73

MMQ Método dos mínimos quadrados. 19, 24, 34, 40

MSE Mean squared error. 64

ReLU Rectified linear unit. 54–56, 60, 62

RMSE Root mean squared error. 64, 70, 75

RNN Recurrent neural network. 7, 61

SARIMA Seasonal autoregressive integrated moving average. 7, 9, 38, 69, 73

SVR Support vector regression. 1, 7, 13, 48, 49, 63, 69, 71–73

XGBoost Extreme gradient boosting. 1, 7, 44, 49, 63, 69, 71–73

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação e apresentação do problema

No dia 1 de Julho de 2020, foram propostos vários temas aos alunos do Mestrado em Matemática e Aplicações pelo Prof. Doutor Eugénio Rocha, Diretor do Mestrado em Matemática e Aplicações. Feita a reflexão mediante os vários temas propostos e após clarificação por parte do Prof. Doutor Eugénio Rocha, decidi acolher este trabalho, em regime de projeto. A escolha deste trabalho em regime de projeto foi consequência da minha vontade de trabalhar em algo que estivesse no limbo entre o teórico e o prático.
Este projeto, que foi inicialmente proposto pela Bosch Termotecnologia-Aveiro, tem como intuito prever o número de vendas de determinadas peças de substituição no sentido de otimizar a produção. Atendendo ao contexto do problema, os clientes não podem esperar pela produção de determinada peça de substituição, pois é preciso que esteja imediamente disponível para entrega. Qual o motivo? Estamos perante peças imprescindíveis, sem as quais, equipamentos de termotecnologia de primeira necessidade não funcionam. Em função disto, o cliente final solicita o envio de peças de substituição, para resolver casos de avaria ou garantir revisões periódicas.

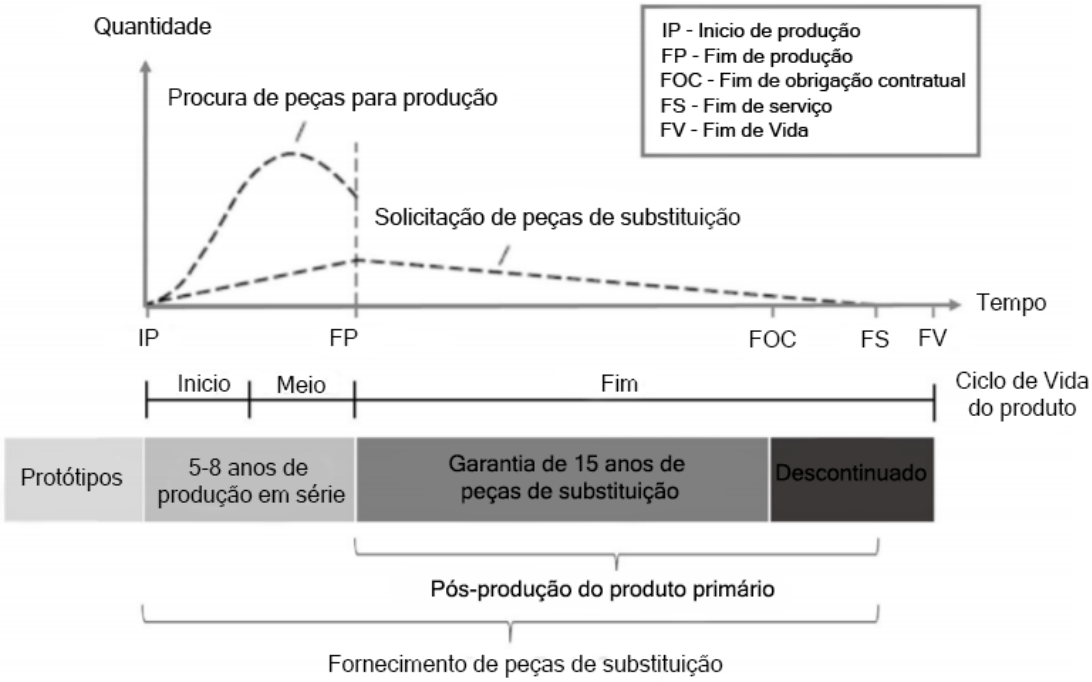

Figura 1.1: Ciclo de vida do produto.
Fonte:[37].

Atualmente, a Bosch já tem um sistema implementado no SAP para prever as vendas de cada peça de substituição, no entanto, é muito básico, uma vez que o número de vendas do ano vigente é uma simples cópia do que ocorreu no ano anterior. Posto isto, o objetivo deste projeto é construir um sistema informático, com base no conhecimento de séries temporais, que consiga prever o número de vendas de cada peça de substituição e, idealmente, melhore as previsões do modelo que a Bosch utiliza. Por forma a acomodar eventuais erros inerentes a previsões por defeito, a Bosch determina um valor de *stock* de segurança para cada peça de substituição.
Os dados em bruto deste projeto vêm num ficheiro .csv com quatro campos: referência do

material, quantidade, entrado em e classe da peça de substituição. As vendas são registadas de forma contínua, pelo que num só dia podem registar-se vários valores diferentes no campo quantidade (número de peças de substituição vendidas) para a mesma peça de substituição. O campo da classe da peça de substituição é determinado pela Bosch, este campo toma três valores possíveis: A, B ou C. As peças de substituição das classes A, B, C representam 80%, 15% e 5% da receita, respetivamente. Klug [37], refere que esta classificação segue o princípio de Pareto (ver [url:33]) e curva de Lorenz(ver [url:34]). Esta atribuição por classe é fundamental, uma vez que, se por um lado pretendemos otimizar a gestão de *stock* através de previsões veiculadas por modelos matemáticos que serão apresentados nesta dissertação, é sabido que o treino de alguns destes modelos é computacionalmente exigente, não compensando afinar os múltiplos parâmetros desses modelos em peças de substituição cuja receita é baixa. Segundo a Bosch, do total de peças de substituição, cerca de 20% têm classificação A sendo responsáveis por 80% da receita, pelo que o foco deste projeto irá recair sob as peças de substituição com classificação A. A atribuição de uma classe a determinada peça de substituição é responsabilidade da Bosch e não está apenas relacionada com o preço de venda, mas também com a quantidade de vendas registadas e com o custo de produção. A Figura 1.2 esquematiza o fluxo deste projeto.

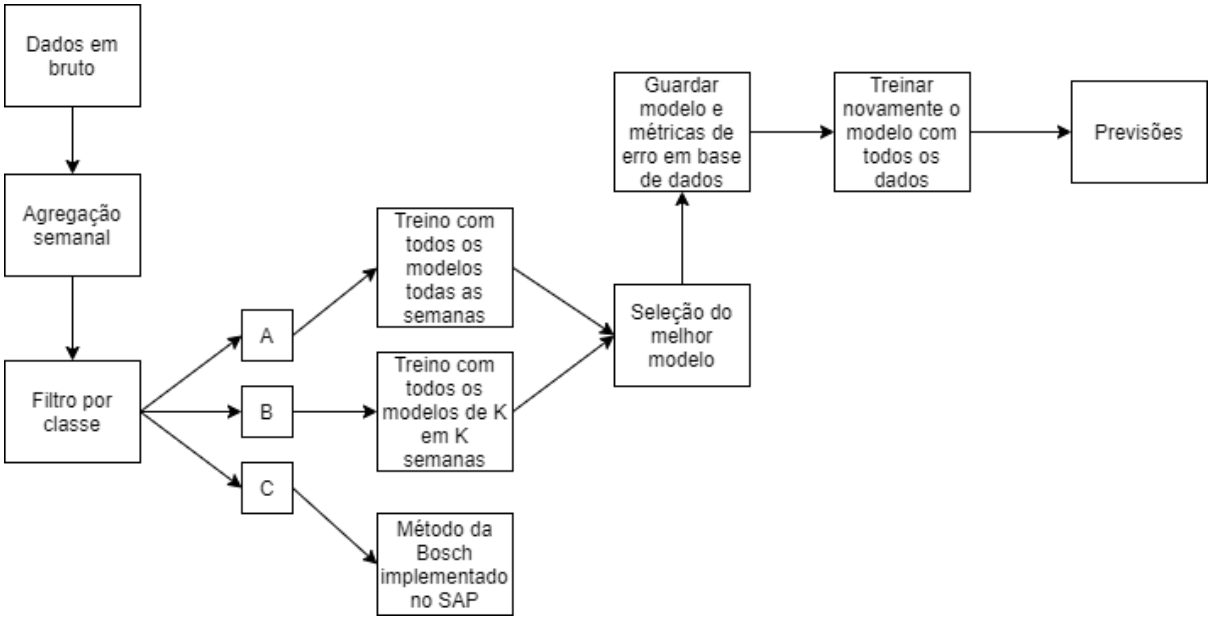

Figura 1.2: Esquema geral do trabalho.

Conforme apresentado na Figura 1.2, este trabalho não se dedicará a fazer previsões em peças de substituição com classificação C, pois entendemos que a despesa resultante do esforço computacional necessário para treinar modelos nestas peças de substituição é superior aos potenciais lucros obtidos. Para as peças de substituição de classe A vamos experimentar todos

os modelos todas as semanas, selecionando o modelo cuja previsão tiver erro inferior. Por último, para as peças de substituição de classe B só experimentamos todos os modelos de K em K semanas, com K definido pela Bosch, sendo que nas semanas intermédias, em função da chegada de novos dados, acertamos os parâmetros e possívelmente hiperparâmetros do modelo anteriormente selecionado.
Os modelos conferem previsões para as próximas três semanas, tempo necessário para a Bosch obter um plano próximo aos consumos que vão ter, ou seja, ajuda a garantir os recursos de pessoas e componentes mais próximo da realidade, pelo que, por questões de coerência, se utiliza para teste as últimas três semanas. Após seleção do melhor modelo segundo determinada métrica de erro, voltamos a treinar o modelo incluindo as últimas três semanas que foram utilizadas na fase de teste.

1.2 Trabalhos relacionados

A otimização da gestão de *stock* é um tema de amplo interesse, naturalmente explicado pelo potencial enorme no aumento das receitas das empresas. A título de exemplo, o setor automóvel representa 7% do PIB na União Europeia (fonte:[url:2]). Destes 7%, as vendas de peças de substituição no setor automóvel têm forte contribuição, dado que representam entre 50% a 70% da receita no pós-venda [37]. Este facto justifica que grande parte das publicações em otimização da gestão de *stock* tenham como caso de estudo, o setor automóvel. Apesar do setor automóvel ser um dos mais interessados na otimização da gestão de *stock*, evidentemente que muitos outros setores têm a mesma pretensão, como é o caso do setor da aviação. A propósito, a Portugália Airlines, subsidiária da TAP, também está interessada em otimizar a gestão de *stock* [36]. J.A Ribeiro [36] aborda as metodologias clássicas e suas variantes, bem como o trabalho realizado pela Raquel Costa em 2015 com o intuito de prever as vendas de perfis de alumínio para a empresa Extrusal, sediada em Aveiro [38]. A dissertação de Raquel Costa explora ainda a metodologia de *bootstrap* (ver [url:25]) aplicada aos resíduos das séries temporais, a mesma abordagem é considerada por Costa et al. [39], embora tenha efeito no modelo Holt-Winters e não no ARIMA. Ainda no setor da aviação, o trabalho concebido por Wang et al. [40] para além dos métodos clássicos utiliza também SVR e um modelo baseado na distribuição de Poisson. Wang et al. [40], abordam ainda uma forma original para seleção dos melhores modelos, com base em regras de associação obtidas através de um algoritmo de aprendizagem não supervisionada para encontrar conjuntos frequentes denominado de *A Priori*. Vargas e Cortes [41] comparam o ARIMA com redes neuronais e modelos híbridos resultantes da conjugação de ambos. Modelos com base na distribuição de Rayleigh e Weibull são explorados em [42]. Após revisão da literatura, constatámos que a maior parte das abordagens utilizadas restringem-se às abordagens clássicas e pequenas variantes, i.e., depreciam a

utilização de modelos de aprendizagem supervisionada, nomeadamente redes neuronais. Este projeto visa também preencher essa lacuna, motivando para a exploração destas novas abordagens, que apesar de computacionalmente mais exigentes, por vezes conferem modelos cujas previsões têm pouco erro. A propósito, Bhuvana et al. [54] abordaram um caso de estudo para previsão de vendas esporádicas de peças de substituição onde o MLP supera métodos clássicos como o ARIMA. Recomendamos ainda a consulta da dissertação de Mestrado de Robby Henkelmann que evidencia o poder das redes neuronais no campo das séries temporais, tendo como caso de estudo o setor automóvel [35].

1.3 Sobre a Bosch

A Bosch foi fundada em 1886 por Robert Bosch na cidade de Estugarda, quando abriu a sua oficina de mecânica de precisão e engenharia elétrica. Em 1897, através da instalação de aparelhos de ignição por magneto para automóveis, a Bosch tornou-se na fornecedora de referência de ignições em virtude da sua alta fiabilidade. Em 1902, esta solução foi melhorada, ignição por magneto de alta-voltagem com vela de ignição, pelo engenheiro chefe, Gottlob Honold.

Figura 1.3: Primeira fábrica da Bosch, construída em 1901, altamente avançada para a época. Fonte:[url:1].

Por volta de 1910, a Bosch já estava implantada em todos os continentes e, em 1913, as vendas fora da Alemanha representavam 88% do total. Seguiram-se tempos tumultuosos por causa da primeira guerra mundial (1914-1918). Neste período, a Bosch teve de reconfigurar a sua produção, deixando de produzir as célebres ignições por magneto para dar lugar a detonadores de granadas. Após a primeira guerra, a Bosch teve a sua vida complicada com várias patentes revogadas, mas continuou com as suas inovações na área automóvel aliadas, claro, à

alta fiabilidade dos equipamentos.

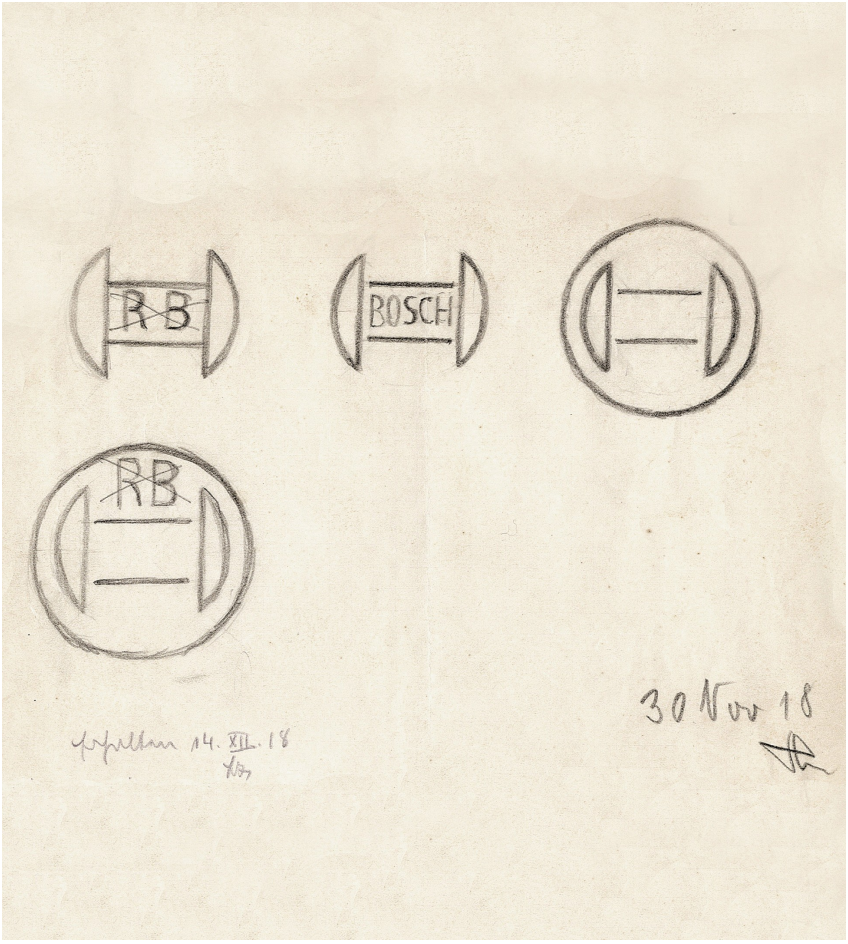

Figura 1.4: Logo da Bosch desenhado por Gottlob Honold em 1918, que permanece até aos dias de hoje.

Fonte:[url:3].

A partir de 1926, em resposta à crise do setor automóvel, a Bosch voltou-se para a produção em série de outros produtos: ferramentas elétricas, equipamentos de termotecnologia, tecnologia de rádio e televisão. Apesar dos esforços, em 1932, as vendas fora da Alemanha representavam 55% do total de vendas, longe do marco atingido em 1913. O futuro não augurava boas notícias para a Bosch pois, em 1939, iria começar outra guerra mundial que obrigou a Bosch a redefinir novamente a sua produção até 1945. Em 1942, o fundador Robert Bosch falece, não assistindo à aniquilação de várias unidades de produção por parte dos Aliados.

Figura 1.5: Estrutura da primeira fábrica da Bosch gravemente danificada, em 1945, após bombardeamento dos Aliados.

Fonte:[url:4].

Após a segunda guerra mundial, as vendas no estrangeiro eram praticamente zero e vários bens extramuros foram expropriados. Sucedeu-se uma fase de reconstrução, com Hans Walz ao comando, em que a Bosch se voltou para a produção de bens domésticos fundamentais, nomeadamente eletrodomésticos de cozinha, berbequins elétricos, autorrádios, etc. A partir de 1950, a Bosch voltou a crescer e, em 1960, já contava com 70.000 colaboradores. Na década de 60, as várias áreas de negócio tornaram-se cada vez mais independentes e, na década de 80, a Bosch entrou no ramo das telecomunicações. Em 1990, com o aparecimento de *software*, surgem novos desafios. Nesta altura, as vendas no estrangeiro representavam 51% do total. Em 2000, as vendas no estrangeiro chegaram a 72%, em parte, consequência da expansão para leste permitida pela queda da cortina de ferro, que separava a Europa Ocidental da Oriental. Atualmente, a Bosch conta com mais de 395.000 colaboradores a nível mundial e atua em quatro setores de negócio: soluções de mobilidade, tecnologia industrial, bens de consumo e tecnologia de energia e edifícios. Ao longo do tempo existiram vários reptos aos quais a Bosch teve de reagir. O paradigma vigente da Bosch é a indústria 4.0 no qual este projeto se enquadra. A Bosch Termotecnologia, em parceria estratégica com a Universidade de Aveiro, está a desenvolver um projeto de inovação direcionado para o setor da indústria, que tem como foco principal o desenvolvimento e implementação de novas tecnologias relacionadas com a indústria 4.0. O leitor interessado pode encontrar mais informação acerca da história da Bosch em [url:5].

1.4 Estrutura da dissertação

O segundo capítulo começa com uma introdução, onde o leitor sem conhecimentos em séries temporais, após leitura, ficará familiarizado com os conceitos gerais e nomenclatura utilizada em séries temporais de forma a poder acompanhar o resto do capítulo. De seguida, à luz do que foi feito no projeto, falamos em técnicas para detetar picos em séries temporais e a forma de os tratar. A restante parte do segundo capítulo detalha a matemática que suporta cada modelo com base em literatura selecionada. O terceiro capítulo aborda pormenores técnicos relativos à implementação da solução. O quarto capítulo expõe os resultados obtidos neste caso de estudo. O último capítulo compreende uma análise crítica aos resultados obtidos, alude para as potencialidades e limitações do trabalho desenvolvido, trabalho futuro e considerações finais. As referências literárias e os endereços web que vão sendo citados ao longo do texto estão disponibilizados no capítulo Bibliografia e Lista de Endereços Web, respetivamente.

Capítulo 2

Séries Temporais e Previsão

2.1 Introdução

Nos últimos anos, tem-se constatado um interesse crescente na área de estatística e *machine learning* em quase todos os setores de atividade. Portanto, é imperativo saber trabalhar com vários tipos de *datasets*. Em muitos casos, os dados são registados ao longo do tempo, o que conduz, naturalmente, à necessidade do estudo das séries temporais.
Informalmente, uma série temporal é uma sequência de observações ao longo do tempo. Usualmente, estas observações são discretas e encontram-se igualmente espaçadas. Todavia, também existem séries temporais contínuas, que não serão foco deste capítulo. Em problemas de regressão, a ordem pela qual as observações ocorrem é indiferente, em séries temporais a ordem é fundamental. O conceito de série temporal decorre naturalmente da noção de processo estocástico, que passamos a definir.

Definição 2.1.1 (Processo estocástico)**.** *Dado o espaço de probabilidade* (Ω, F, P)*, onde* Ω *representa o espaço amostral (conjunto de saídas possíveis),* F *o conjunto de eventos,* P *uma função de probabilidade que calcula a probabilidade de determinado evento ocorrer e um conjunto* T*, que pode ser discreto ou contínuo dependendo da natureza do processo, um processo estocástico é uma função* $X(t, \omega)$ *definida no produto cartesiano* $T \times \Omega$*. Pode-se escrever como* $\{X_t, t \in T\}$*.*

O conceito de processo estocástico é uma generalização de variável aleatória, dado que, para cada $t \in T$ obtemos uma variável aleatória. Outra forma de olhar para um processo estocástico é como uma família de variáveis aleatórias ordenadas.
Conforme informámos, apenas vamos trabalhar com séries temporais em tempo discreto com observações igualmente espaçadas, portanto o conjunto T é discreto. Doravante, por simplificação, quando nos referirmos a uma série temporal ou processo estocástico utilizaremos a notação $\{X_t\}$, embora, implicitamente, nos estejamos a referir a $\{X_t, t \in T\}$.

2.1.1 Tendência

Seja $\{X_t\}$ uma série temporal com $X_t = m_t + Y_t$ onde m_t representa a componente tendencial e Y_t a componente aleatória (não explicada por m_t) com média zero. Uma forma clássica de estimar m_t é via Método dos mínimos quadrados (MMQ). O intuito é utilizar o MMQ para ajustar os dados a uma família paramétrica de funções, e.g.,

$$m_t = a_0 + a_1 t. \tag{2.1}$$

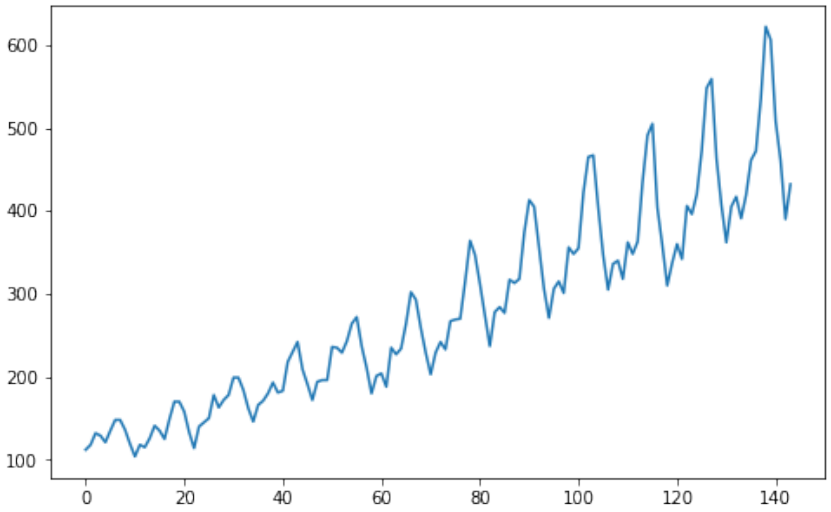

Figura 2.1: Número de passageiros aéreos (em milhares) por mês.
Dataset:[url:6].

A série temporal da Figura 2.1, claramente denota uma tendência de crescimento linear. Como tal, considerando a componente m_t, conforme proposto em (2.1), obtemos as estimativas de a_0 e a_1 via MMQ,

$$a_0 = 90.310 \text{ e } a_1 = 2.657.$$

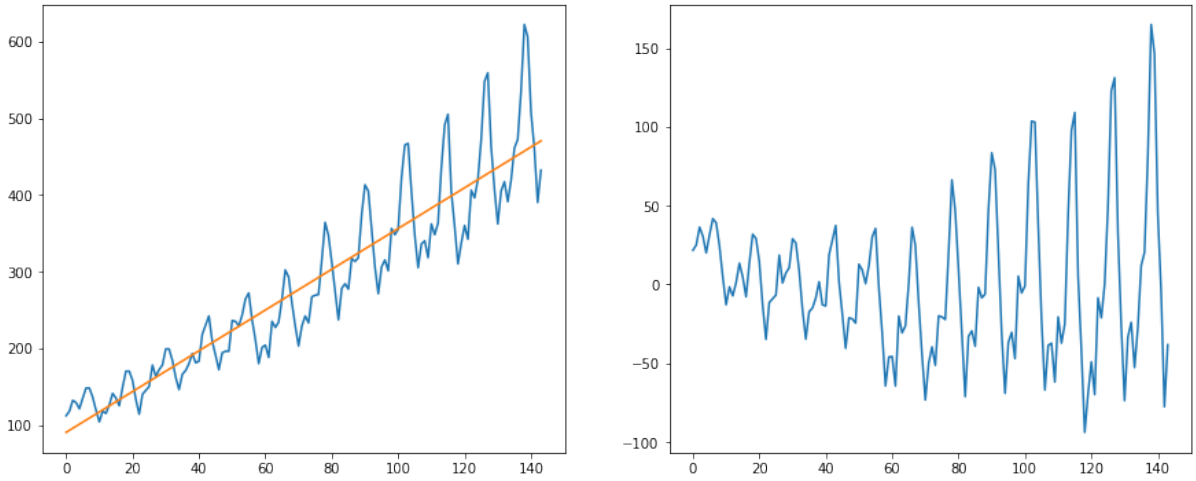

Figura 2.2: Regressão linear à esquerda, resíduos à direita.
Dataset: [url:6].

2.1.2 Sazonalidade

Em algumas séries temporais existe uma forte componente sazonal, como é o caso da Figura 2.3 (esquerda). Existem várias formas de extrair a componente sazonal de uma série temporal, o método aqui utilizado segue os preceitos utilizados em processamento de sinal [4]. Brockell e Davis [1], utilizam um método alternativo (regressão harmónica). Considerando a decomposição: $X_t = s_t + Y_t$ onde s_t representa a componente sazonal e Y_t a componente aleatória, não explicada por s_t. Uma escolha natural para s_t é ser representada pela série de Fourier. Assim, temos a expansão,

$$s(t) = \sum_{k=0}^{K} a_k cos(2\pi kft) + \sum_{k=0}^{K} b_k sin(2\pi kft) = \sum_{k=0}^{K} A_k cos(2\pi kft + \phi_k), \tag{2.2}$$

onde os coeficientes são determinados por,

$$a_k = \frac{2}{T}\int_0^T cos(2\pi kft)\ dt \qquad b_k = \frac{2}{T}\int_0^T sin(\pi kft)\ dt,$$

$$A_k = \sqrt{b_k^2 + a_k^2} \qquad \phi_k = -\arctan\left(\frac{b_k}{a_k}\right).$$

Aplicando a fórmula de Euler à expressão (2.2), resulta em

$$s(t) = \sum_{k=0}^{K} A_k cos(2\pi kft + \phi_k) = \sum_{k=0}^{K} A_k \frac{e^{j(2\pi kft+\phi_k)} + e^{-j(2\pi kft+\phi_k)}}{2} =$$

$$\frac{1}{2}\left(\sum_{k=0}^{K} A_k e^{j\phi_k} e^{j2\pi kft} + \sum_{k=0}^{K} A_k e^{-j\phi_k} e^{-j2\pi kft}\right).$$

Tomando

$$C_k = A_k e^{j\phi_k} + A_k e^{-j\phi_k},$$

obtemos a série exponencial de Fourier

$$s(t) = \sum_{k=-K}^{K} C_k e^{j2\pi kft}, \quad C_k \in \mathbb{C}. \tag{2.3}$$

Na prática, a componente s_t é discreta, o que nos conduz à Discrete Fourier transform (DFT). O Algoritmo Fast Fourier transform (FFT) aplica a DFT de forma eficiente. Utilizando uma frequência de amostragem ligeiramente superior ao dobro da frequência mais elevada (critério de Nyquist) e reconstruíndo a série, podemos obter a componente sazonal da série temporal, ver Figura 2.3 (meio).

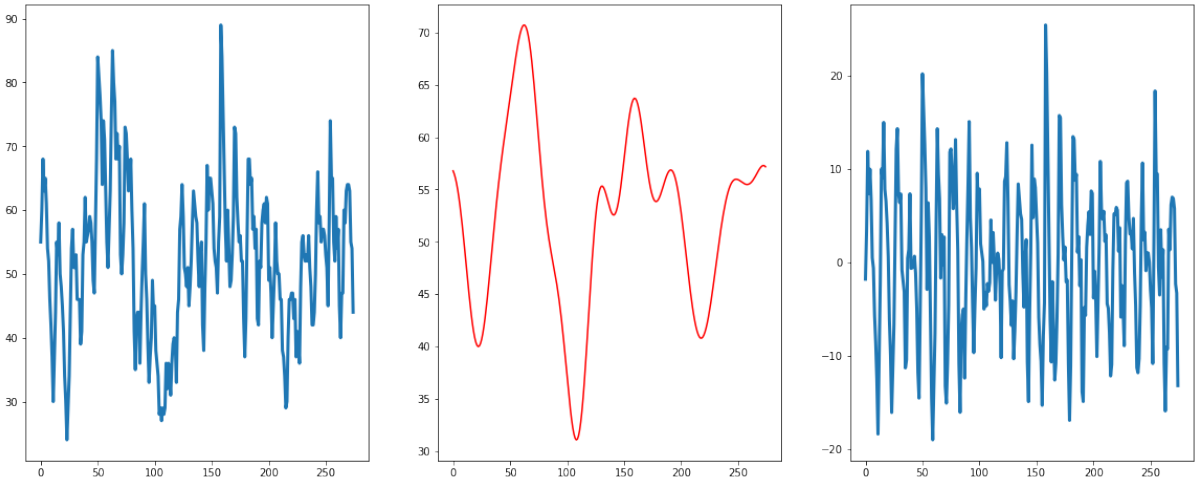

Figura 2.3: Série temporal original à esquerda , componente sazonal no meio, resíduos à direita.

Dataset: [url:7].

2.1.3 Estacionariedade

De forma informal, uma série temporal diz-se estacionária se as propriedades estatísticas de $\{X_t\}$ são semelhantes às de $\{X_{t+h}\}$ para qualquer h inteiro. Outra regra prática elenca que, numa série temporal estacionária, características sumárias como a média e variância são consistentes ao longo do tempo.

Definição 2.1.2 (Ruído branco)**.** *Um processo estocástico* $\{X_t\}$ *diz-se que é um ruído branco,* $RB(0, \sigma^2)$*, com* $\sigma^2 > 0$*, caso se cumpram as condições:*

- $\forall t \in T, \quad E[X_t] = 0$;
- $\forall t \in T, \quad Var[X_t] = \sigma^2 < +\infty$;
- $\forall s, t \in T : s \neq t \implies Cov(X_s, X_t) = 0$.

Definição 2.1.3 (Função de autocovariância)**.** *Seja $\{X_t\}$ um processo estocástico com $T = \mathbb{Z}$. A função autocovariância, é dada por*

$$\gamma(h) = Cov(X_t, X_{t+h}) = Cov(X_{t+h}, X_{t+h-h}) = \gamma(-h), \quad \forall h \in \mathbb{Z}$$
$$\textit{Caso particular: } \gamma(0) = Var(X_t).$$

Definição 2.1.4 (Função de autocorrelação)**.** *Seja $\{X_t\}$ uma processo estocástico com $T = \mathbb{Z}$. A função autocorrelação, é dada por*
$\rho(-h) = Corr(X_{t+h}, X_{t+h-h}) = Corr(X_{t+h}, X_t) = \rho(h) = \frac{Cov(X_{t+h}, X_t)}{\sqrt{Var(X_{t+h})Var(X_t)}} = \frac{\gamma(h)}{\gamma(0)}, \quad \forall h \in \mathbb{Z}$

Definição 2.1.5 (Processo estocástico fortemente estacionário)**.** *Um processo estocástico $\{X_t\}$ diz-se fortemente estacionário se*

$$(X_{t_1}, ..., X_{t_n}) \overset{d}{=} (X_{t_1+h}, ..., X_{t_n+h}),\ h > 0,$$

onde o símbolo $\overset{d}{=}$ significa que as distribuições conjuntas de ambos os membros são iguais. Como a verificação da estacionaridade forte é, em geral, difícil, é mais comum verificar-se a noção de estacionaridade fraca.

Definição 2.1.6 (Processo estocástico fracamente estacionário)**.** *Um processo estocástico $\{X_t\}$ diz-se fracamente estacionário, se $\forall t \in T$:*

- $E[X_t] = \mu$ *(não depende de t);*
- $E[X_t^2] < \infty$;
- $Cov(X_{t+h}, X_t) = \gamma(h)$ *(é independente de t para cada h).*

Doravamente, dir-se-á que uma série temporal é estacionária, caso cumpra as condições relativas à estacionariedade fraca.
Nas duas secções anteriores deste capítulo explorámos as noções de componente sazonal e componente tendencial. Braei e Wagner [5] fazem ainda alusão a uma outra componente, a componente cíclica, flutuações não regulares em torno da tendência, ver Figura 2.4, e também à componente do nível, que será utilizada na construção do modelo de Holt-Winters. A componente do nível é simplesmente a média da série temporal, portanto se a série temporal tem componente tendencial vincada, o nível vai-se alterando.

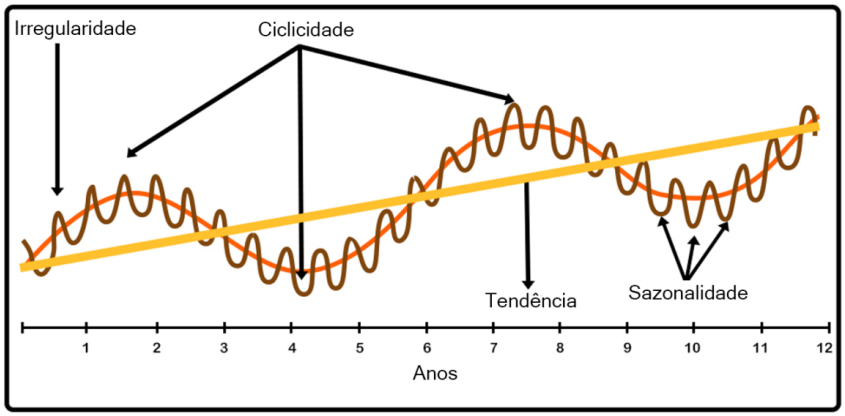

Figura 2.4: Componentes de uma série temporal.
Fonte:[url:8].

Definição 2.1.7 (Modelo aditivo)**.** $X_t = m_t + s_t + c_t + Y_t$ *onde* m_t *representa a componente tendencial,* s_t *a componente sazonal,* c_t *a componente cíclica e* Y_t *os resíduos.*

Definição 2.1.8 (Modelo multiplicativo)**.** $X_t = m_t \times s_t \times c_t \times Y_t$ *onde* m_t *representa a componente tendencial,* s_t *a componente sazonal,* c_t *a componente cíclica e* Y_t *os resíduos.*

Para tornar uma série temporal não estacionária em estacionária, primeiramente devemos representá-la graficamente para verificar a existência ou não das componentes tendencial, sazonal e cíclica. De seguida, deve-se estimar as componentes relevantes e subtrair (modelo aditivo) ou dividir (modelo multiplicativo) as componentes à série original. A Figura 2.3 (direita) demonstra a série temporal dos resíduos, obtida após subtração da componente sazonal, que, para efeitos práticos, se considera estacionária. Outras técnicas baseiam-se na série das diferenças. Substitui-se a série temporal orignal $\{X_t\}$ por $\{Y_t := X_t - X_{t-d}\}$ para algum inteiro positivo d, usualmente, d=1.

2.1.4 Testes de estacionariedade

Esta subsecção tem forte influência dos trabalhos [1, 6, 11, 14].

Após aplicação de uma das duas metodologias propostas na secção anterior, estamos interessados em perceber, se a série temporal dos resíduos é estacionária. Existem vários procedimentos práticos para detetar se uma série temporal é estacionária, que passamos a explanar:

- Visualização gráfica da série temporal;
- Estudar as características sumárias da série temporal ao longo do tempo;
- Testes de hipóteses.

Esta secção aborda a terceira possibilidade.

Testes de raízes unitárias

Fuller, em 1976, foi quem introduziu os testes de raízes unitárias [10], capítulo 10. De 1979 a 1981 Dickey e Fuller criaram os testes de raízes unitárias, que ficaram conhecidos na literatura como testes de Dickey-Fuller [14, 43]. Posteriormente, desenvolveram-se outros testes de raízes unitárias, como por exemplo, o Teste De Phillips-Perron e Teste KPSS, mas que não serão considerados neste trabalho [12, 13].

Teste de Dickey-Fuller

Considere-se o processo estocástico AR(1)

$$X_t = \phi_1 X_{t-1} + Y_t, \tag{2.4}$$

onde Y_t é um ruído branco (processo aleatório com média zero e variância limitada). O polinómio autorregressivo do processo AR(1) é $\phi_1(B) = 1 - \phi_1 B$. O processo estocástico é estacionário quando as raízes do polinómio autorregressivo estão fora do círculo unitário, Teorema 2.3.1, quer isto dizer que para o processo ser estacionário, o módulo das raízes do polinómio autorregressivo, possivelmente complexas, deve ser superior a 1. Considerando a equação característica $1-\phi_1 z = 0$ associada ao polinómio autorregressivo, obtém-se a raíz $z^* = \frac{1}{\phi_1}$. Logo, para o processo ser estacionário é necessário que $|z^*| > 1$, equivalente a $|\phi_1| < 1$. Calculando a primeira diferença no processo estocástico AR(1) obtemos,

$$X_t - X_{t-1} = \phi_1 X_{t-1} + Y_t - X_{t-1}. \tag{2.5}$$

Como $\Delta X_t = X_t - X_{t-1}$ e fazendo $\delta = \phi_1 - 1$ temos a equação de regressão,

$$\Delta X_t = \delta X_{t-1} + Y_t. \tag{2.6}$$

Desta forma, o nosso teste de hipóteses para a estacionariedade toma a forma,

$$H_0 : \delta = 0 \text{ (equivalente a } \phi_1 = 1) \quad vs \quad H_1 : \delta < 0 \text{ (equivalente a } \phi_1 < 1). \tag{2.7}$$

Seja $\hat{\delta}$ a estimativa via MMQ de δ com recurso à equação 2.6. Desta forma, $\hat{\phi}_1 = 1 + \hat{\delta}$ é a estimativa via MMQ de ϕ_1. Para n suficientemente grande, temos que

$$\sqrt{n}(\phi_1 - \hat{\phi}_1) \overset{a}{\sim} N(0, \sqrt{1 - \phi_1^2}). \tag{2.8}$$

No entanto, Evans e Sans [15], p.763, demonstraram que a aproximação à distribuição normal, equação 2.8, não é razoável, mesmo para valores elevados de n. Evans e Sans demonstra-

ram ainda que a distribuição do parâmetro autorregressivo ϕ_1, sob H_0 ($\phi_1 = 1$), é enviesada à esquerda, com recurso aos métodos de simulação de Monte Carlo [15, 44]. Em virtude dos problemas elencados supra, é mais comum utilizar-se uma distribuição específica τ. Os valores de τ_{crit}, que dependem de n (dimensão da amostra) e α (nível de significância), podem ser encontrados numa tabela de Dickey-Fuller.

Teste de Dickey-Fuller aumentado

O modelo de regressão utilizado na equação (2.6) é o mais simples, não inclui constante nem componente tendencial. Existem outros dois modelos de regressão,

$$\Delta X_t = \delta X_{t-1} + \beta_1 + Y_t \quad \text{(Constante)} \tag{2.9}$$

$$\Delta X_t = \delta X_{t-1}\beta_1 + \beta_2 t + Y_t \quad \text{(Constante e tendência)} \tag{2.10}$$

O modelo de regressão utilizado no Teste de Dickey-Fuller aumentado utiliza diferienciação até ao *lag* m, e pode ser aplicado a qualquer um dos três modelos de regressão. Note-se que, nos modelos de regressão das equações anteriores m=1. Assim, diferenciando até ao *lag* m o modelo de regressão com constante e tendência, vem que

$$\Delta X_t = \beta_1 + \beta_2 t + \delta X_{t-1} + \sum_{i=1}^{m} \alpha_i \Delta X_{t-i} + Y_t, \tag{2.11}$$

onde δ é o coeficiente de presença de raiz unitária, β_1 é a constante, β_2 é o coeficiente da tendência e m é o número de *lags* da série temporal. Para uma dedução completa da equação 2.11, consultar [45], capítulo 2. O valor m deve ser escolhido por forma a que os resíduos da série temporal não sejam correlacionados de forma significativa, i.e., os resíduos seguem um ruído branco. Técnicas possíveis para escolher m incluem a minimização do Akaike's information criterion (AIC) (ver [url:26]) ou Bayesian information criterion (BIC) (ver [url:27]). Relembrando o teste de hipóteses,

$$H_0 : \delta = 0 \quad vs \quad H_1 : \delta < 0, \tag{2.12}$$

o teste de Dickey-Fuller aumentado tem como estatística de teste,

$$\tau_{obs} = \frac{\hat{\delta}}{se(\hat{\delta})} \tag{2.13}$$

onde $\hat{\delta}$ é uma estimativa de δ e se($\hat{\delta}$) a estimativa do desvio padrão do erro de δ. Os valores τ_{crit} encontram-se numa tabela de Dickey-Fuller.

Acabamos esta secção com um exemplo ilustrativo. Conforme se pode constatar visualmente (ver Figura 2.5), para um nível de significância de $\alpha = 0.05$, apenas obtemos estacionariedade

após diferenciar a série temporal dos passageiros áereos duas vezes (ver Tabela 2.1).

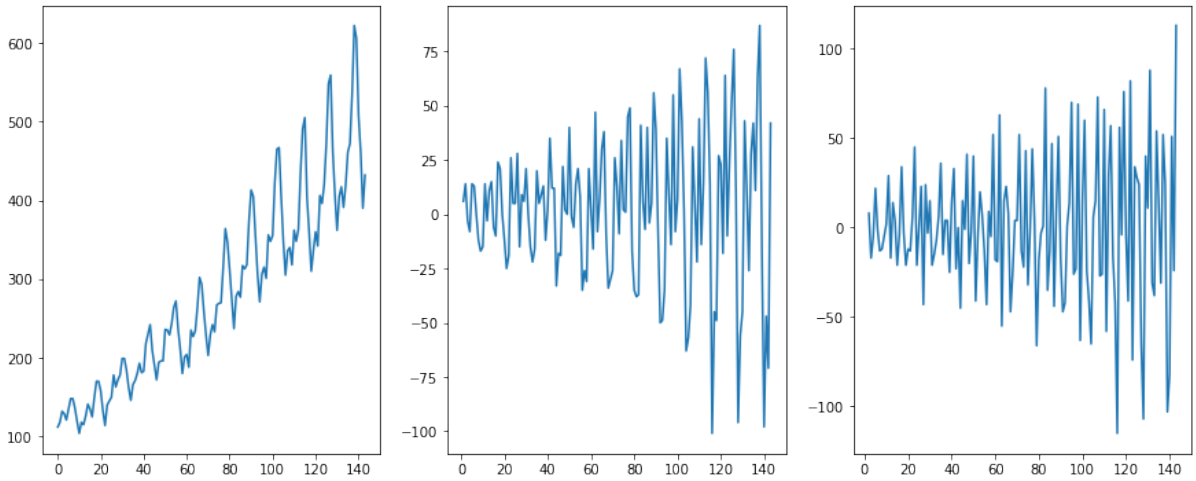

Figura 2.5: Série temporal original (passageiros aéreos em milhares) à esquerda, primeira diferença no meio, segunda diferença à direita.

Dataset:[url:6].

Série Temporal	p-value
Série Original	0.992
1ª Série das diferenças	0.054
2ª Série das diferenças	2.733e-29

Tabela 2.1: Resultados do teste de Dickey-Fuller para $\alpha = 0.05$

2.1.5 Testes para os resíduos

O estudo dos resíduos obtidos após diferenciação da série temporal ou subtração/divisão das compomentes é essencial. Os testes que se seguem permitem que possamos perceber se existe dependência linear entre as observações. Note-se que, existem ligeiras diferenças entre a FAC e a FACP. Enquanto que, a FAC mensura a correlação entre os valores de uma série temporal e os seus valores prévios (*lags*), a FACP não contabiliza o efeito na correlação das observações intermédias. Assim, é expectável que a FAC e a FACP estejam consonantes para o primeiro *lag*, conforme se pode verificar na Figura 2.6, mas possívelmente diferentes[1] para os próximos *lags*.

[1]Suponha-se que uma série temporal tem correlação de 0.8 entre $\{X_t\}$ e $\{X_{t-1}\}$, bem como entre $\{X_{t-1}\}$ e $\{X_{t-2}\}$. Assim, tanto a FAC como a FACP terão o mesmo valor para o primeiro *lag*. No entanto, para o segundo *lag* a FAC apresentará um valor de 0.8*0.8=0.64 e a FACP um valor de zero, à exceção de um erro aleatório.

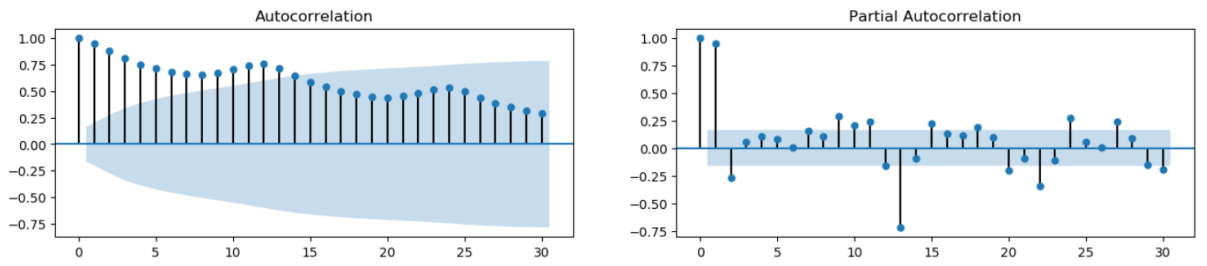

Figura 2.6: FAC à esquerda, FACP à direita. Ambas para 30 *lags*.
Dataset:[url:6]

Testes com base no correlograma

Um teste simples baseia-se na FAC para o *lag* k definido por

$$\rho(k) = \frac{\gamma(k)}{\gamma(0)}. \tag{2.14}$$

É obvio que $\rho(0) = 1$ e $-1 \leq \rho(k) \leq 1$ para os restantes valores de k inteiros. Bartlett [7] demonstrou que para n suficientemente grande a FAC de uma sequência de resíduos i.i.d. $Y_1, Y_2, ..., Y_n$ pode ser aproximada pela distribuição N(0, $\frac{1}{n}$). Posto isto, se $Y_1, Y_2, ..., Y_n$ forem resíduos i.i.d. é de esperar que (1-α)% dos valores da FAC amostral estejam no intervalo $[-\frac{Z_{1-\frac{\alpha}{2}}}{\sqrt{n}}, \frac{Z_{1-\frac{\alpha}{2}}}{\sqrt{n}}]$.

Teste de Box-Pierce

Sob as mesmas condições do teste com base no correlograma, Box e Pierce [8] desenvolveram um teste com as seguintes hipóteses:

$$H_0 : \rho(k) = 0 \text{ para todo k} \in \{0, 1, ..., n\} \quad H_1 : \text{ existe um k} \in \{0, 1, ..., n\} \text{ tal que } \rho(k) \neq 0 \tag{2.15}$$

Segue-se a estatística de teste Q proposta por Box e Pierce, sob hipótese de H_0 para n suficientemente grande,

$$Q = n \sum_{k=1}^{n} \rho(\hat{k})^2 \overset{a}{\sim} \chi^2_n, \tag{2.16}$$

onde $\overset{a}{\sim}$ significa que a estatística Q segue assintoticamente, i.e., quando $n \to \infty$, a distribuição χ^2_n. Como $\rho(k) \overset{a}{\sim} N(0, \frac{1}{n}) \implies \sqrt{n}\rho(k) \overset{a}{\sim} N(0, 1) \implies n\rho(k)^2 \overset{a}{\sim} \chi^2_1 \implies Q \overset{a}{\sim} \chi^2_n$. Para um nível de significância α, se $Q > \chi^2_{1-\alpha}$ a hipótese nula (H_0) é rejeitada.

Teste de Ljung-Box

Ljung e Box [9] propuseram uma alteração ao teste de Box-Pierce, que apresenta maior potência[2] para amostras pequenas,

$$Q_{LB} = n(n+1)\sum_{k=1}^{n}\frac{\rho(\hat{k})^2}{n-k} \overset{a}{\sim} \chi^2_n. \tag{2.17}$$

2.2 Pré-processamento

Antes de abordar os vários modelos de previsão para séries temporais, é aconselhável explorar e filtrar os dados. Nesta secção, vamos dedicar-nos ao estudo de picos. A noção de pico, aqui, não significa apenas um máximo local, mas uma observação com valor excessivamente alto relativamente à sua vizinhança, em função de um valor de excesso predeterminado. Note-se que, a noção de pico aqui descrita distingue-se do conceito de *outlier* (observação que difere significativamente das restantes). Um pico é também um *outlier*, mas nem todos *outliers* são picos, pois o conceito de *outlier* também engloba observações com valor excessivamente baixo relativamente às restantes. Para além disso, o conceito de pico aqui definido é num sentido circunscrito, conforme sugere a equação 2.18. A Figura 2.7 mostra uma série temporal com, pelo menos, um pico.

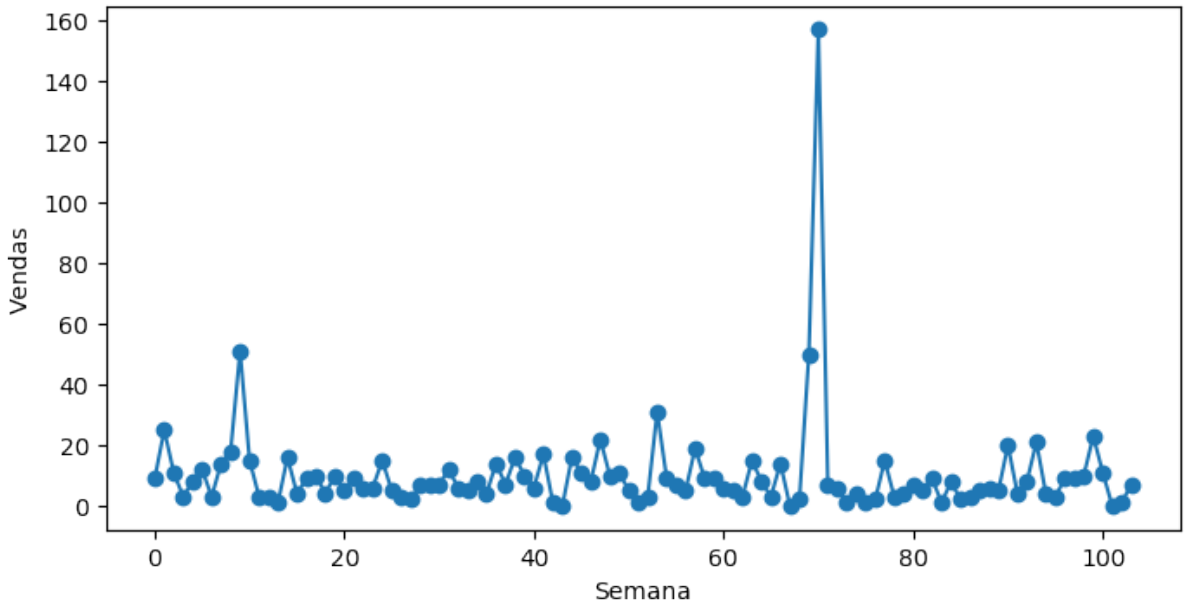

Figura 2.7: Série temporal de uma peça de substituição com, pelo menos, um pico.

[2]A potência estatística é a probabilidade de que o teste rejeite corretamente a hipótese nula (H_0) quando uma hipótese alternativa (H_1) é verdadeira. Assim, quanto maior for a potência, menor é a probabilidade de cometer um erro do tipo 2.

2.2.1 Deteção de picos

Pelo limite superior do modelo ARIMA

Sejam $x_1, ..., x_n$ valores de uma série temporal e $f^{\alpha}(x_{i-k}, x_{i-k+1}, ..., x_{i-1})$ uma função que prevê um limite superior, x_i^{sup} de $x_i \in (x_{k+1}, ..., x_n)$ com base nos k *lags* anteriores para um nível de significância $\alpha \in]0, 1[$. Assim, se $x_i > x_i^{sup}$, então x_i é pico. A Figura 2.8 é um exemplo de aplicação.

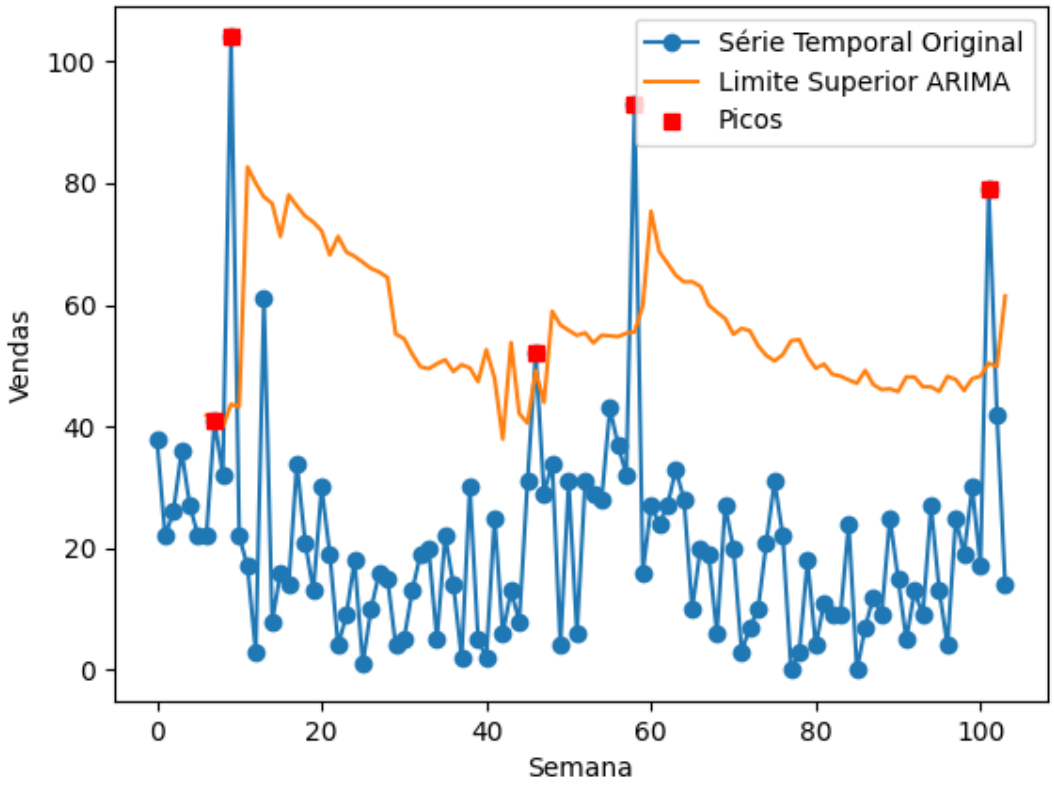

Figura 2.8: Série temporal de uma peça de substituição com o limite superior ARIMA para $\alpha = 0.05$.

Algoritmo de Palshikar modificado

Este algoritmo, conforme o nome indica, é uma ligeira alteração ao algoritmo proposto por Palshikar em 2009 [16]. Sejam $x_1, ..., x_n$ valores de uma série temporal, $k \in \mathbb{N}$ e S uma função que atribuí um *score* a cada observação x_i, com $i \in \{k+1, ..., n-k-1\}$, para um determinado

k definida por

$$S(x_i, k) = \frac{\max\limits_{1\leq j\leq k}\left\{\frac{x_i - x_{i-j}}{x_i}\right\} + \max\limits_{1\leq j\leq k}\left\{\frac{x_i - x_{i+j}}{x_i}\right\}}{2}. \tag{2.18}$$

No caso de existir algum $x_i = 0$, a função S toma a forma

$$S(x_i, k) = \frac{\max\limits_{1\leq j\leq k}\{x_i - x_{i-j}\} + \max\limits_{1\leq j\leq k}\{x_i - x_{i+j}\}}{2}. \tag{2.19}$$

A função S gera uma série temporal de *scores* $\{S(x_i, k)\}_{k+1\leq i\leq n-k-1}$. Sejam μ_s e σ_s a média e desvio padrão da série temporal de *scores*, respetivamente. Se $s_i \equiv S(x_i, k) > \mu_s + \alpha\sigma_s$, então x_i é pico. Usualmente, escolhe-se $\alpha \in [2, 3]$. A Figura 2.9 mostra um exemplo de aplicação do algoritmo.

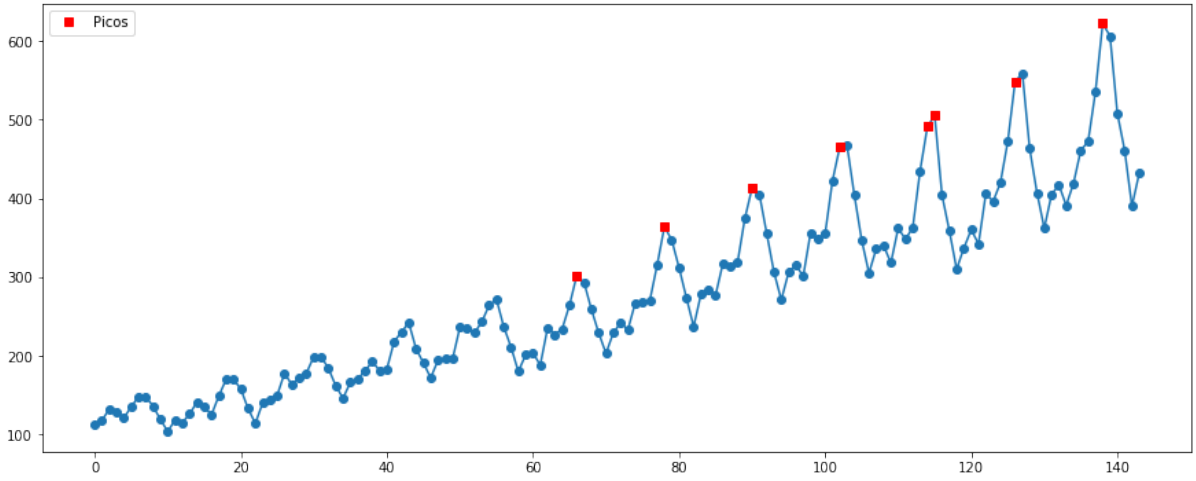

Figura 2.9: Série temporal dos passageiros aéreos com os picos detetados pelo Algoritmo de Girish Palshikar modificado para k=5 e $\alpha = 2$.

2.2.2 Tratamento de picos

Em problemas de regressão, é comum remover os *outliers* antes de treinar um modelo. Tal abordagem não é possível em séries temporais devido à dependência temporal que existe entre as observações. Posto isto, a técnica clássica passa por utilizar um dos modelos desta dissertação que melhor se ajuste aos dados e substituir o pico pela previsão dada pelo modelo, sendo que os picos não são utilizados para treino do modelo. Após a deteção de cada pico, utiliza-se para treino do modelo todas as observações que o antecedem e substitui-se este pela previsão do modelo utilizado.

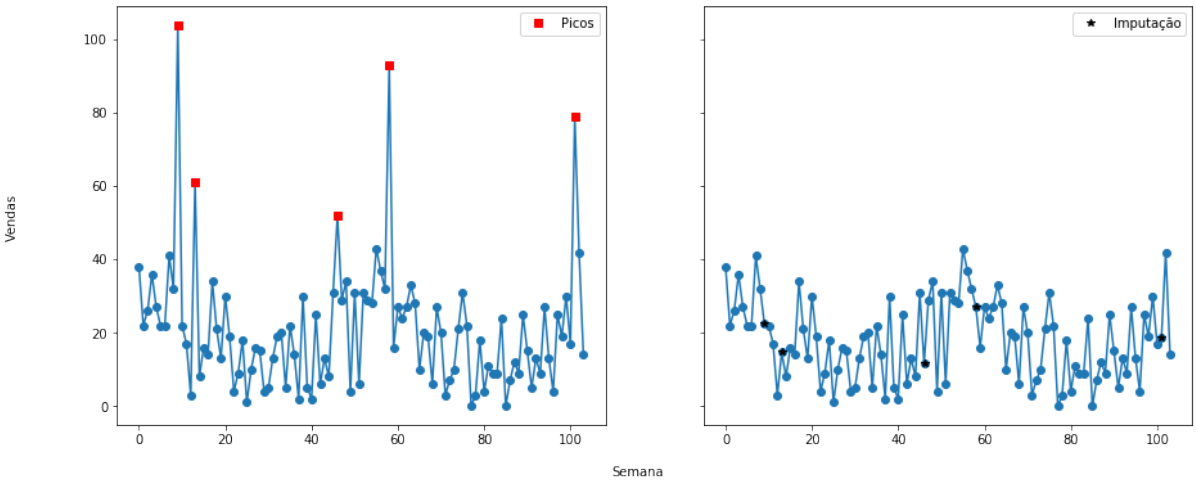

Figura 2.10: Série temporal original de uma peça de substituição com deteção de picos via algoritmo de Palshikar modificado à esquerda, série temporal com imputação via ARIMA à direita.

2.3 Métodos clássicos

Esta secção visa fazer um apanhado dos modelos de previsão mais utilizados em séries temporais, que são os modelos autorregressivos e de alisamento exponencial. Estes modelos são os mais frequentemente utilizados devido à sua facilidade de implementação, no entanto, modelos autorregressivos como o ARIMA e suas variantes são pouco flexíveis, dado que não conseguem detetar relações não lineares entre o valor a prever e os seus *lags*, para além de que exigem que a série temporal seja estacionária.

2.3.1 ARIMA

Conforme foi dito, os modelos autorregressivos exigem que a série temporal seja estacionária, portanto pressupõe-se aqui que a série temporal $\{X_t\}$ é estacionária ou foi tornada estacionária por um dos métodos da secção introdutória (remoção das componentes ou diferenciação). Comecemos pelo processo autorregressivo. Formalmente, um modelo autorregressivo AR(p) de $\{X_t\}$ pode ser escrito na forma compacta, através do seu polinómio autorregressivo, como $\phi_p(B)X_t = Y_t$ onde $Y_t \sim RB(0, \sigma^2)$ e $\phi_p(B) = 1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^p$ com $\phi_i \in \mathbb{R}$ para i $\in \{1, ..., p\}$. Note-se que, $B^k X_t = X_{t-k}$. Alternativamente, pode-se escrever AR(p) na forma extensa

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} + Y_t, \tag{2.20}$$

com $\phi_i \in \mathbb{R}$ para i $\in \{1, ..., p\}$.
Murteira et al. [24], p.57, apresentam um Teorema que dá uma condição necessária para que

um modelo autorregressivo seja estacionário através das raízes do seu polinómio autorregressivo. Na prática é uma condição necessária e suficiente, mas só faremos a prova no primeiro sentido. Segue-se o Teorema com a prova.

Teorema 2.3.1. *Seja* $\phi_p(z) = 1 - \phi_1 z - \phi_2 z^2 - ... - \phi_p z^p$ *um polinómio de grau p tal que* $\phi_p(z) \neq 0$ *para todo o* $|z| \leq 1$*, i.e., não tem raízes no círculo unitário. Assim,* $[\phi_p(z)]^{-}1$ *tem um desenvolvimento em série de potências da forma:*

$$[\phi_p(z)]^{-1} = \sum_{j=0}^{\infty} \omega_j z^j$$

onde,

$$\sum_{j=0}^{\infty} |\omega_j| \leq \infty.$$

Demonstração. Temos que

$$\sum_{j=0}^{\infty} r^j = (1-r)^{-1} \quad \text{para } |r| < 1. \tag{2.21}$$

A equação (2.21) mostra que o inverso de um polinómio de grau 1 pode ser escrito como um polinómio de grau infinito. Fatorizando $\phi_p(z)$,

$$\phi_p(z) = \prod_{i=1}^{p} \left(1 - \frac{z}{r_i}\right) \tag{2.22}$$

onde $r_1, r_2, ..., r_p$ são as raízes do polinómio. Utilizando as equações 2.21 e 2.22 para o argumento $|z| \leq min(|r_i|)$, temos que $|\frac{z}{r_i}| < 1$ para $i \in \{1, 2, ..., p\}$. Desta forma obtemos

$$[\phi_p(z)]^{-1} = \prod_{i=1}^{p} \left(1 - \frac{z}{r_i}\right)^{-1} = \prod_{i=1}^{p} \left(\sum_{j=0}^{\infty} \left(\frac{z}{r_i}\right)^j\right) = \sum_{j=0}^{\infty} \omega_j z^j,$$

onde os coeficientes w_j decaem para zero quando j$\rightarrow \infty$. □

Resulta deste teorema, que para um processo AR(p)

$$\phi_p(B) X_t = Y_t \implies X_t = [\phi_p(B)]^{-1} Y_t = \sum_{j=0}^{\infty} \omega_j Y_{t-j}, \tag{2.23}$$

verificando-se a estacionariedade do processo.
Um processo estocástico MA(q) com $q < \infty$ é composto pela soma de vários ruídos brancos, conforme veremos, pelo que é sempre estacionário. Logo, se a um processo autorregressivo

com polinómio autorregressivo nas condições do Teorema 2.3.1 for acrescentado um processo estocástico MA(q) com $q < \infty$, continua a ser estacionário. Atendendo a este argumento, para estudar a estacionariedade de qualquer processo desta secção basta verificar as raízes do polinómio autorregressivo.

Estimação de parâmetros AR(p) via equações de Yule-Walker

Considere-se um modelo AR(p) da série temporal $\{X_t\}$ com $\mu = E[X_t]$. Se $\mu \neq 0$, reescrever X_t como $\hat{X}_t = X_t - \mu$ obtendo-se $\hat{\mu} = E[\hat{X}_t] = 0$. Logo, vamos assumir que $\mu = 0$ à partida. Utilizando a função de autocovariância, atendendo a que $E[X_t] = 0$ e à estacionariedade da série temporal $\{X_t\}$, vem que

$$\gamma(k) = Cov(X_t, X_{t+k}) = Cov(X_t, X_{t-k}) = E[X_t X_{t-k}] \tag{2.24}$$

$$= \phi_1\gamma(k-1) + \phi_2\gamma(k-2) + ... + \phi_p\gamma(k-p), \tag{2.25}$$

passando da equação (2.24) para a equação (2.25) pela aplicação equação (2.20).
Dividindo a equação (2.25) por $\gamma(0)$ obtemos

$$\rho(k) = \phi_1\rho(k-1) + \phi_2\rho(k-2) + ... + \phi_p\rho(k-p). \tag{2.26}$$

A divisão por $\gamma(0)$ é legítima, pois $\gamma(0) = Var[X_t] > Var[Y_t] = \sigma^2 > 0$.
De seguida, fazendo k variar em $\{1, 2, ..., p\}$ na equação (2.26), obtém-se o sistema

$$\begin{aligned}
&\phi_1 + \phi_2\rho(1) + ... + \phi_p\rho(p-1) = \rho(1) \quad \text{(k=1)}\\
&\phi_1\rho(1) + \phi_2 + ... + \phi_p\rho(p-2) = \rho(2) \quad \text{(k=2)}\\
&\qquad\qquad \cdots\\
&\phi_1\rho(p-1) + \phi_2\rho(p-2) + ... + \phi_p = \rho(p) \quad \text{(k=p)}.
\end{aligned}$$

As equações anteriores são denominadas como equações de Yule-Walker, que reescritas na forma matricial, vem que

$$\begin{bmatrix}
1 & \rho(1) & \rho(2) & \dots & \rho(p-1)\\
\rho(1) & 1 & \rho(2) & \dots & \rho(p-2)\\
\rho(2) & \rho(1) & 1 & \dots & \rho(p-3)\\
 & & \cdots & & \\
\rho(p-1) & \rho(p-2) & \rho(p-3) & \dots & 1
\end{bmatrix}
\begin{bmatrix}
\phi_1\\ \phi_2\\ \phi_3\\ \cdots\\ \phi_p
\end{bmatrix}
=
\begin{bmatrix}
\rho(1)\\ \rho(2)\\ \rho(3)\\ \cdots\\ \rho(p)
\end{bmatrix} \tag{2.27}$$

O valores de $\rho(k)$ para $k \in \{1, ..., p\}$ são dados pela FAC amostral. Os coeficientes $\phi_1, ..., \phi_p$ podem ser determinados pela Regra de Cramer, ou através de métodos numéricos iterativos,

como Gauss–Seidel.

Estimação de parâmetros AR(p) via MMQ

Escrevendo AR(p) na forma extensa, ver equação (2.20), constatamos que o problema de estimar $\phi_1, ..., \phi_p$ é semelhante ao problema de regressão linear

$$X_t = (\phi_1, ..., \phi_p) \begin{pmatrix} X_{t-1} \\ \vdots \\ X_{t-p} \end{pmatrix} + Y_t = \boldsymbol{X_{t-1}^T}\phi + Y_t. \tag{2.28}$$

Considerando $\phi = (\phi_1, ..., \phi_p)^T$ e $\boldsymbol{X_{t-1}} = (X_{t-1}, ..., X_{t-p})^T$. A estimativa via MMQ de ϕ é dada por

$$\hat{\phi} = \left(\sum_{t=p+1}^{n} \boldsymbol{X_{t-1}} \boldsymbol{X_{t-1}^T} \right)^{-1} \left(\sum_{t=p+1}^{n} \boldsymbol{X_{t-1}} X_t \right) \tag{2.29}$$

Também é possível estimar os parâmetros via Estimação de máxima verosimilhança (EMV). Com efeito, passa-se a exemplificar para o caso particular AR(1).

Estimação de parâmetros AR(1) via EMV

Considerando o modelo AR(1)

$$X_t = \phi X_{t-1} + Y_t \text{ com } Y_t \sim N(0, \sigma^2),$$

segue que a função densidade de probabilidade de $(Y_2, ..., Y_n)$ é dada por

$$f(Y_2, ..., Y_n) = \left(\frac{1}{2\pi\sigma^2} \right)^{(n-1)/2} \exp\left(-\frac{1}{2\sigma^2} \sum_{t=2}^{n} Y_t^2 \right). \tag{2.30}$$

Como $X_n = \phi X_{n-1} + Y_t$, o determinante da transformação do Jacobiano é 1. Assim,

$$f(X_2, ..., X_n | X_1) = \left(\frac{1}{2\pi\sigma^2} \right)^{(n-1)/2} \exp\left(-\frac{1}{2\sigma^2} \sum_{t=2}^{n} (X_t - \phi X_{t-1})^2 \right). \tag{2.31}$$

Se $Y_1, Y_0, Y_{-1}, Y_{-2}, ...$ são i.i.d., então $X_1 \sim N(0, \frac{\sigma^2}{1-\phi^2})$. Desta forma, a função de máxima verosimilhança é dada por,

$$\begin{aligned} L(\phi) &= f(X_1, ..., X_n) = f(X_2, ..., X_n | X_1) f(X_1) \\ &= \left(\frac{1}{2\pi\sigma^2} \right)^{n/2} \sqrt{1-\phi^2} \exp\left(-\frac{1}{2\sigma^2} \left(\sum_{t=2}^{n} (X_t - \phi X_{t-1})^2 + (1+\phi^2) X_1^2 \right) \right). \end{aligned}$$

A estimativa $\hat{\phi}$ de ϕ é o maximizante da função de máxima verosimilhança L.

Definição 2.3.2 (Processo de médias móveis MA(q))**.** *O processo MA(q) define-se na forma compacta por* $X_t = \theta(B)Y_t$*, com* $\theta(B) = 1 + \theta_1 B + ... + \theta_q B^q$ *e na forma extensa por*

$$X_t = Y_t + \theta_1 Y_{t-1} + ... + \theta_q Y_{t-q} \quad \textit{onde } Y_t \sim RB(0, \sigma^2), \tag{2.32}$$

onde $\theta_i \in \mathbb{R}$ *para* $i \in \{1, ..., q\}$*.*

Estimação parâmetros MA(q)

A estimação dos parâmetros para um processo MA(q) é, em geral, difícil. Vamos ilustrar o processo para o caso particular MA(1), o caso MA(q) para q>1 é uma generalização de q=1. Se $|\theta| < 1$, resulta do processo MA(1) que $Y_t = X_t + \theta Y_{t-1} = \sum_{i=0}^{+\infty} \theta^i X_{t-i}$, pelo que $Y_t \equiv Y_t(\theta)$. Seja $S(\theta) = \sum_{t=1}^{n} Y_t^2$. Podemos determinar $\theta \in]-1, 1[$ de forma a minimizar a função S. Fixando $Y_0 = 0$ e $\theta = \theta^* \in]-1, 1[$, temos

$$\begin{aligned} Y_1 &= X_1, \\ Y_2 &= X_2 + \theta^* Y_1 = X_2 + \theta^* X_1, \\ &\vdots \\ Y_n &= X_n + \theta^* Y_{n-1}. \end{aligned}$$

Este sistema de equações permite calcular $S_*(\theta) = \sum_{t=1}^{n} Y_t^2$ para θ^*. Para determinar o minimizante de $S_*(\theta)$ em $]-1, 1[$, vamos utilizar um método numérico, o método de Gauss-Newton. Este método é também conhecido como *conditional least squares.* Considere-se

$$Y_t(\theta) \approx Y_t(\theta^*) + (\theta - \theta^*) \frac{dY_t(\theta)}{d\theta}|_{\theta=\theta^*} \tag{2.33}$$

Note-se que a equação (2.33) é linear, portanto $S_*(\theta) = \sum_{t=1}^{n} Y_t^2$ pode ser minimizado analiticamente por forma a obter uma nova aproximação $\theta^{(1)}$. De seguida, substitui-se $\theta^{(1)}$ por θ^* na equação (2.33). Este processo é repetido até convergir.

Atente-se que as derivadas $\frac{dY_t(\theta)}{d\theta}|_{\theta=\theta^*}$ são calculadas de forma recursiva

$$\frac{dY_t(\theta)}{d\theta}|_{\theta=\theta^*} = \theta^* \frac{dY_{t-1}(\theta)}{d\theta}|_{\theta=\theta^*} + Y_{t-1}(\theta^*), \quad \frac{dY_0(\theta)}{d\theta} = 0. \tag{2.34}$$

Para o caso geral de determinar os parâmetros do modelo MA(q) para q>1, utiliza-se o método de Gauss-Newton multivariado para minimizar $S_*(\theta)$ com a diferença que, neste caso, $Y_t = X_t + \theta_1 Y_{t-1} + ... + \theta_q Y_{t-q}$ e $\theta = (\theta_1, ..., \theta_q)^T$.

Em alguns casos, a nossa série temporal ajusta-se a um modelo composto por uma junção do processo AR(p) e MA(q). A esse modelo chamamos de ARMA(p,q).

Definição 2.3.3 (Processo ARMA(p,q)). *O processo ARMA(p,q) define-se na forma compacta por* $\phi(B)X_t = \theta(B)Y_t$, *com* $\phi(B) = 1 - \phi_1 B - ... - \phi_p B^p$ *e* $\theta(B) = 1 + \theta_1 B + ... + \theta_q B^q$, *na forma extensa*

$$X_t = \phi_1 X_{t-1} + ... + \phi_p X_{t-p} + Y_t + \theta_1 Y_{t-1} .. + \theta_q Y_{t-q}, \tag{2.35}$$

com $\phi_i, \theta_j \in \mathbb{R}$ *para* $i \in \{1, ..., p\}$ *e* $j \in \{1, ..., q\}$.

Estimação parâmetros ARMA(p,q)

A estimação pode ser obtida pelo método de Gauss-Newton de forma análoga ao processo MA(q), com a diferença que, em vez da função $S_*(\theta)$, temos a função $S_*(\phi, \theta)$= $\sum_{t=1}^{n} Y_t^2(\phi, \theta)$ com $\phi = (\phi_1, ..., \phi_p)^T$ e $\theta = (\theta_1, ..., \theta_q)^T$.

Na secção introdutória vimos que algumas séries temporais não são estacionárias, sendo nesses casos possível diferenciar a série temporal original até potencialmente obter estacionariedade.

Definição 2.3.4 (Processo ARIMA(p,d,q)). *O processo ARIMA(p,d,q) define-se na forma compacta por* $\phi(B)(1 - B)^d X_t = \theta(B)Y_t$. *Usualmente* $d \leq 3$.

Seleção de p e q em modelos ARIMA(p,d,q)

De acordo com o descrito por Murteira et al. [24], p.69, nos processos MA(q) ocorre um decaimento brusco da FAC para k=q+1 e nos processos AR(p) tal decaimento brusco ocorre na FACP para k=p+1. Com base nesta informação, é possível determinar os parâmetros p e q através da FAC e da FACP amostrais.

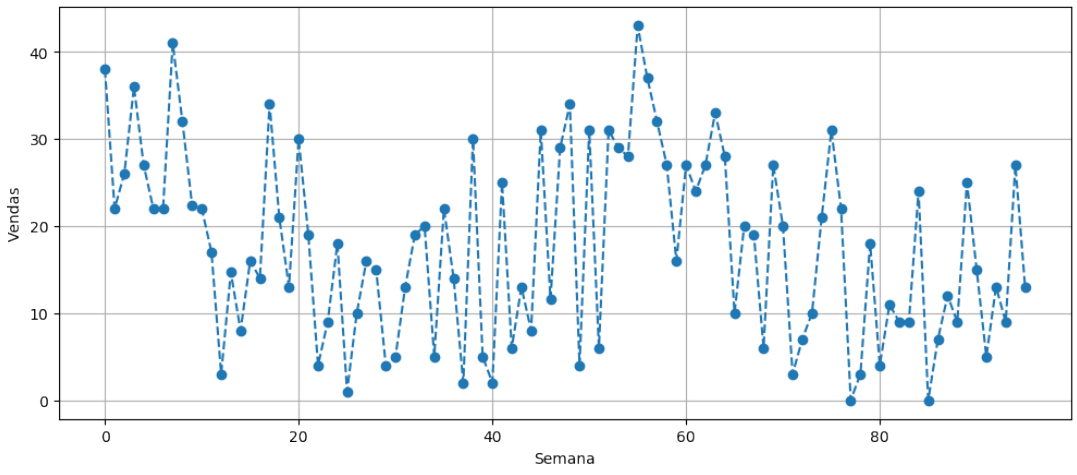

Figura 2.11: Seleção p e q em modelos ARIMA(p,d,q) para uma série temporal de uma peça de substituição.

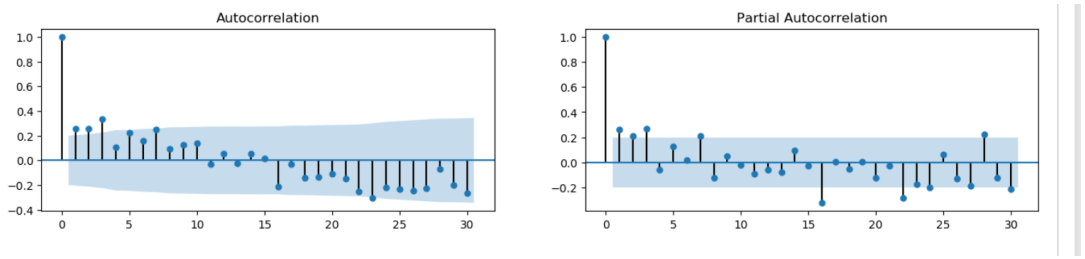

Figura 2.12: FAC e FACP amostrais da série temporal da uma peça de substituição da Figura 2.11.

A série apresenta um p-value $= 0.014$ para o Teste de Dickey-Fuller, pelo que, para um nível de significância de $\alpha = 0.05$, se crê que a série temporal seja estacionária, d=0. Com base na Figura 2.13 conclui-se que p=3 e q=3, uma vez que existem três *lags* significativos na FAC (escolha do q) e na FACP (escolha do p). Logo, escolhemos p=3, q=3 e d=0.

Metodologia de Box-Jenkis para previsão

Seja $\{X_t\}$ um processo modelado por um modelo ARMA(p,q) nas condições do Teorema 2.3.1, então $X_t = \sum_{i=0}^{\infty} \omega_i Y_{t-i}$. De acordo com Shewhart e Wilks [23], pp.73-74, considerando os seguintes valores médios condicionais,

$$E[X_s|X_t, ...] = \begin{cases} X_s, & s \leq t \\ X_{s+t}, & s > t \end{cases} \tag{2.36}$$

$$E[Y_s|X_t, ...] = \begin{cases} Y_s, & s \leq t \\ 0, & s > t \end{cases} \tag{2.37}$$

a previsão de X_{t+h} é dada por

$$\hat{X}_{t+h} = E[X_{t+h}|X_t, ...] = \sum_{i=0}^{\infty} \omega_i E[Y_{t+h-i}|X_t, ...] = \sum_{i=h}^{\infty} \omega_i Y_{t+h-i}. \tag{2.38}$$

Demonstra-se que, esta estimativa através da média condicional de $\hat{X}_{t+h}$, é a melhor em termos de erro quadrático médio [23], p.72. Segue-se que o erro da previsão é dado por

$$e_t(h) = X_{t+h} - \hat{X}_{t+h} = \sum_{i=0}^{h-1} \omega_i Y_{t+h-i}, \tag{2.39}$$

e a variância do erro da previsão dada por:

$$\sigma(e_t(h))^2 = E[e_t(h)^2|X_t, ...] = \sigma_Y^2 \sum_{i=0}^{h-1} \omega_i^2. \tag{2.40}$$

Assumindo que os erros são normalmente distribuídos, é possível determinar uma região de confiança ao nível de significância α por

$$\left[\hat{X}_{t+h} - Z_{\frac{\alpha}{2}}\ \sigma(e_t(h)), \hat{X}_{t+h} + Z_{\frac{\alpha}{2}}\ \sigma(e_t(h))\right]. \tag{2.41}$$

Nos casos em que a nossa série temporal $\{X_t\}$ para além de depender de $X_{t-1}, X_{t-2}, ...$ também depende de $X_{t-s}, X_{t-2s}, ...$, ou seja, apresenta uma componente sazonal, os modelos supra tornam-se inadequados. Nesses casos utilizamos o modelo SARIMA.

Definição 2.3.5 (Processo SARIMA(p,d,q)×(P,D,Q))**.** *O processo SARIMA(p,d,q)×(P,D,Q) define-se na forma compacta por* $\phi(B)\Phi_P(B^s)(1-B)^d(1-B^s)^D X_t = \theta(B)\Theta_Q(B^s)Y_t$ *onde,*

$$\begin{aligned}
\phi(B) &= 1 - \phi_1 B - ... - \phi_p B^p,\\
\theta(B) &= 1 - \theta_1 B - ... - \theta_q B^q,\\
\Phi_P(B^s) &= 1 - \Phi_1 B^s - ... - \Phi_P B^{sP},\\
\Theta_Q(B^s) &= 1 - \Theta B^s - ... - \Theta_Q B^{sQ}.
\end{aligned}$$

Na Figura 2.13 apresenta-se um exemplo de aplicação aos dados de vendas de uma peça de substituição

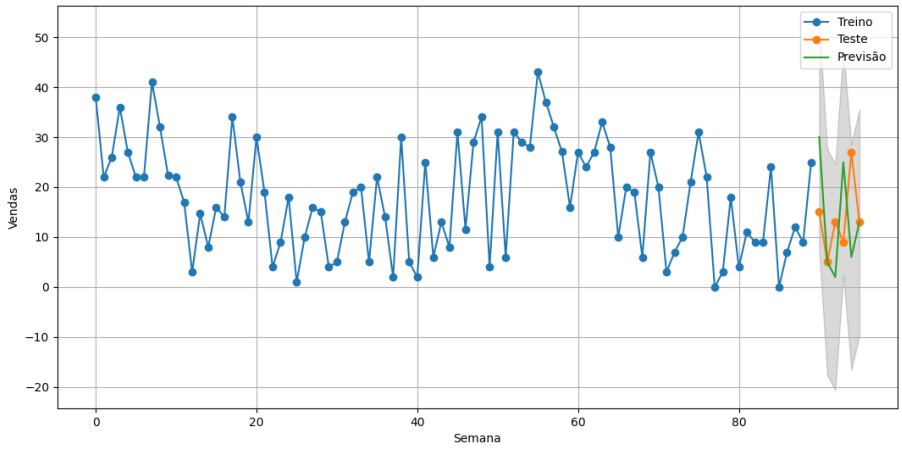

Figura 2.13: Previsão das vendas de uma peça de substituição para as próximas 6 semanas pelo modelo SARIMA.

Esta introdução aos modelos autorregressivos baseou-se em [23, 24].

2.3.2 Holt-Winters

O modelo de previsão de Holt (1957) e Winters (1960) é mais flexível do que o ARIMA, no que concerne aos pré-requisitos. O modelo de Holt-Winters não exige que a série temporal seja estacionária e pode ser apresentado de duas formas: aditiva e multiplicativa. Este modelo pertence à familia de modelos de alisamento exponencial. Pelo facto do modelo ter três componentes, também é conhecido como alisamento exponencial triplo. A variante aditiva do modelo deve ser utilizada quando as variações na sazonalidade são constantes ao longo do tempo, enquanto que a variante multiplicativa é a recomendada quando a variação na sazonalidade é proporcional ao nível da série temporal. Recomenda-se que a série temporal tenha um mínimo de dois períodos de sazonalidade, para que seja ajustada devidamente ao modelo de Holt-Winters.

Holt-Winters aditivo

As equações do modelo aditivo são as seguintes

$$l_t = \alpha(X_t - s_{t-p}) + (1-\alpha)(l_{t-1} + m_{t-1}), \tag{2.42}$$

$$m_t = \beta(l_t - l_{t-1}) + (1-\beta)m_{t-1}, \tag{2.43}$$

$$s_t = \gamma(X_t - l_t) + (1-\gamma)s_{t-p}, \tag{2.44}$$

onde l_t, m_t, s_t representam o nível, tendência e sazonalidade, respetivamente. Os parâmetros de alisamento devem respeitar a condição $0 \leq \alpha, \beta, \gamma \leq 1$, p representa o período de sazonalidade. Naturalmente, se dividirmos o ano em meses, temos p=12, se agregarmos à semana p=52 e temos p=365 para uma agregação diária. A previsão para um próximo passo h é dada por

$$\hat{X}_{t+h} = l_t + hm_t + s_{t-p-h_p^+}, \tag{2.45}$$

com $h_p^+ = \lfloor (h-1) \bmod \mathrm{p} \rfloor + 1$. Considerando uma série temporal $\{X_t\}$ os valores iniciais são dados por

$$l_0 = \frac{1}{p}(X_1 + X_2 + ... + X_p), \tag{2.46}$$

$$m_0 = \frac{1}{p}\left(\frac{X_{p+1} - X_1}{p} + ... + \frac{X_{2p} - X_p}{p}\right), \tag{2.47}$$

$$s_0 = X_p - l_0, s_{-1} = X_{p-1} - l_0, ..., s_{-p+1} = X_1 - l_0. \tag{2.48}$$

A componente do nível é inicializada pela média aritmética da série temporal, a componente tendencial é inicializada à custa da média das derivadas discretas, e a componente sazonal é obtida à custa da diferença entre o valor da série temporal e o valor do nível.

Taylor e Letham [52], membros da equipa de *data science* do Facebook, criaram um modelo aditivo denominado por FBProphet que tem uma componente irregular para lidar com eventos e utiliza modelos complexos e flexíveis para estimar a componente tendencial, [52], pp.8-11.

Holt-Winters multiplicativo

As equações do modelo multiplicativo são dadas por

$$l_t = \alpha \frac{X_t}{s_p} + (1-\alpha)(l_{t-1} + m_{t-1}), \tag{2.49}$$

$$m_t = \beta(l_t - l_{t-1}) + (1-\beta)m_{t-1}, \tag{2.50}$$

$$s_t = \gamma \frac{X_t}{l_{t-1} - m_{t-1}} + (1-\gamma)s_{t-p}.. \tag{2.51}$$

A previsão para um passo h

$$\hat{X}_{t+h} = (l_t + hm_t)s_{t-p-h_p^+}, \tag{2.52}$$

onde a inicialização é idêntica ao modelo aditivo, simplesmente trocando na componente sazonal a operação de diferença pela operação de divisão.

Otimização dos parâmetros de alisamento

Para ambos os modelos, aditivo e multiplicativo, os parâmetros α, β e γ podem ser obtidos via MMQ. Supondo um *dataset* com N tuplos, pretendemos minimizar o erro quadrático

$$E(\alpha, \beta, \gamma) = \sum_{t=3}^{N} (\hat{X}_t - X_t)^2, \tag{2.53}$$

com $\hat{X}_t$ a previsão dependente das componentes (ver (2.45) e (2.52))

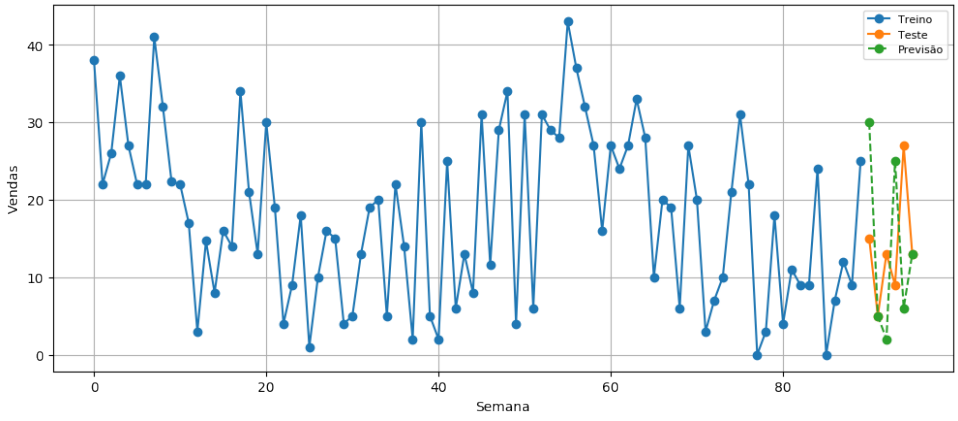

Figura 2.14: Previsão das vendas de uma peça de substituição para as próximas 6 semanas pelo modelo Holt-Winters.

Para mais informação sobre métodos de alisamento exponencial, recomendamos ver [25].

2.4 Métodos de aprendizagem supervisionada

Segundo Bronwlee [3], p.14, na aprendizagem supervisionada utilizamos algoritmos para aprender uma função f que mapeia as variáveis de *input* X$\in \mathbb{R}^n$, para as variáveis de *output* Y$\in \mathbb{R}^m$, i.e.

$$Y = f(X)$$

Usualmente, nas aplicações $n \geq 2$ e m=1. Neste caso, usaremos y para denotar Y_1.
Para utilizar algoritmos de aprendizagem supervisionada em séries temporais é necessário reestruturar os nossos dados como um problema de aprendizagem supervisionada com recurso a uma janela deslizante (cf. Figura 2.15).

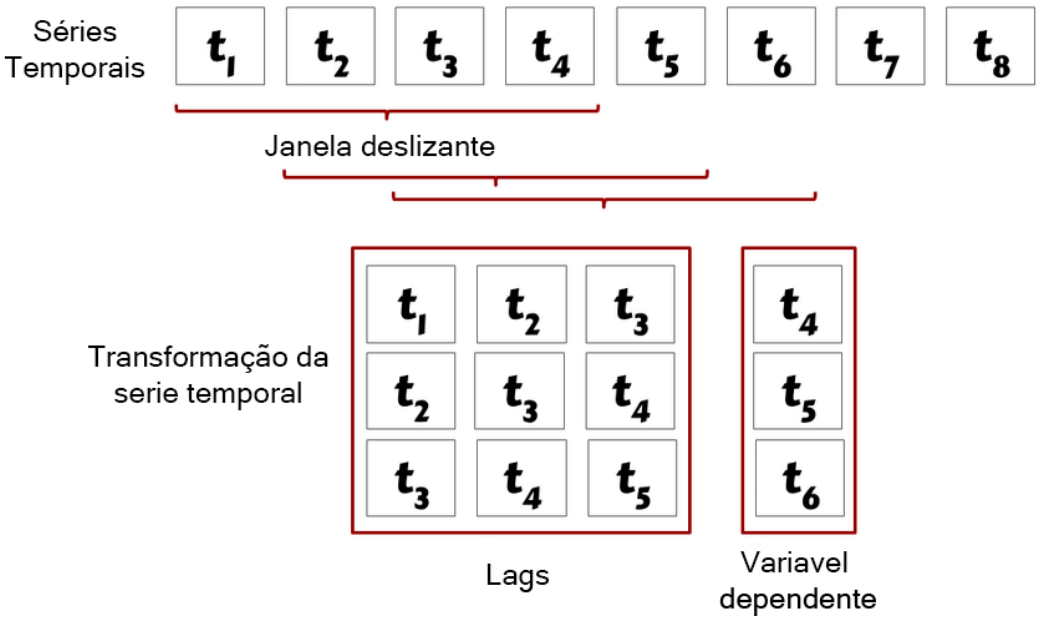

Figura 2.15: Série temporal como um problema de aprendizagem supervisionada.
Fonte: [url:9].

Validação Cruzada

A validação cruzada é essencial para a otimização dos hiperparâmetros dos modelos desta secção. Devido à dependência temporal das séries temporais não se utilizam os métodos de validação cruzada clássicos como *k-fold cross-validation*, ver [url:10]). A Figura 2.16 mostra de que forma o *k-fold cross-validation* pode ser adaptado para séries temporais.

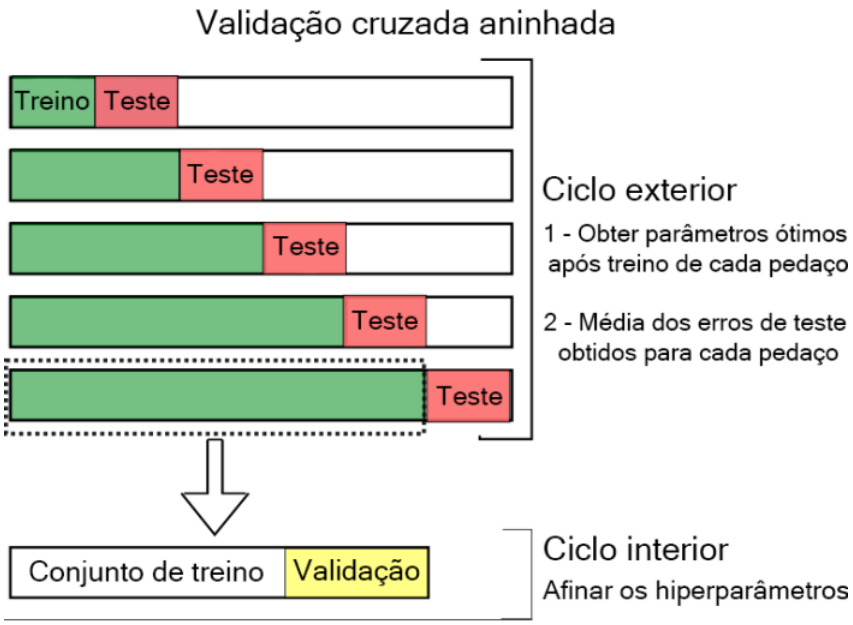

Figura 2.16: Validação cruzada aninhada.
Fonte: [url:11].

2.4.1 XGBoost

XGBoost é uma versão com regularização das *Gradient Boosting Trees*, ver [url:12]. Para além da regularização, existem outras melhorias:

1. Apresenta robustez ao lidar com valores omissos;

2. Utiliza estruturas de dados eficientes;

3. Adapta-se melhor ao processamento em paralelo, reduzindo o tempo de treino.

A pretensão desta secção é dissecar a intuição matemática do algoritmo proposto por Chen e Guestrin em 2016 [11]. Suponha-se $\{(x_n, t_n)\}_{n\in\{1,..,N\}}$ um *dataset*, com $t_n \in \mathbb{R}$ a variável dependente de $x_n \in \mathbb{R}^D$, que se pretende prever. Antes de apresentar a fórmula utilizada para previsão, necessitamos de introduzir a família de funções $\mathcal{F}$ definida por $\mathcal{F} = \{f(x) = w_q(x)\}$ com q: $\mathbb{R}^D \to T$, uma função que mapeia um tuplo para o índice da folha da árvore de regressão, sendo T o número de folhas da árvore de regressão. Posto isto, estamos em condições de apresentar a fórmula para previsão, que é dada por

$$y_n = \phi(x_n) = \sum_{k=1}^{K} f_k(x_n), \quad f_k \in \mathcal{F}. \tag{2.54}$$

Conforme podemos constatar pela equação (2.54), este algoritmo é um algoritmo de *Ensemble learning* (ver [url:13]), sendo a previsão y_n resultante da adição dos *scores* de K árvores de regressão. O método baseia-se em minimizar a função objetivo

$$\mathcal{L}(\phi) = \sum_{n=1}^{N} E(t_n, y_n) + \sum_{k=1}^{K} \Omega(f_k), \tag{2.55}$$

onde E é uma função diferenciável e convexa (ver [19], p.19) que mede o erro entre a previsão y_n e o valor real t_n, e $\Omega(f_k) = \gamma T + \frac{1}{2}\lambda||w||^2$, um termo de regularização do modelo, por forma a evitar *overfitting*. Diz-se que existe *overfitting* quando um determinado modelo se ajusta quase perfeitamente a um determinado *dataset*, mas comporta-se terrivelmente para novos dados.

Não é possível determinar o minimizante de $\mathcal{L}(\phi)$ através de métodos de otimização tradicionais no espaço euclidiano, uma vez que o modelo de *ensemble* da equação 2.55 inclui funções como parâmetros. Por este motivo, utilizamos uma abordagem *greedy* para minimizar $\mathcal{L}$. Seja $y_n^{(t)}$ a previsão na t-ésima iteração, a abordagem *greedy* minimiza,

$$\mathcal{L}^{(t)} = \sum_{n=1}^{N} E(t_n, y_n^{(t-1)} + f_t(x_n)) + \Omega(f_t). \tag{2.56}$$

Significa isto que adicionamos de forma *greedy* a função $f_t \in \mathcal{F}$ que melhor minimiza a função objetivo $\mathcal{L}(\phi)$.

Podemos aproximar $\mathcal{L}^{(t)}$ com recurso à expansão em série de Taylor até à segunda ordem, obtendo assim um problema quadrático,

$$\mathcal{L}^{(t)} \approx \sum_{n=1}^{N} \left(E\left(t_n, y_n^{(t-1)}\right) + g_n f_t(x_n) + \frac{1}{2} h_n f_t(x_n)^2 \right) + \Omega(f_t), \tag{2.57}$$

com $g_n = \frac{\partial E(t_n, y_n^{(t-1)})}{\partial y_n^{(t-1)}}$ e $h_n = \frac{\partial^2 E(t_n, y_n^{(t-1)})}{\partial^2 y_n^{((t-1)}}$. Dado que $\sum_{n=1}^{N} E(t_n, y_n^{(t-1)})$ é uma constante, afeta o mínimo, mas não afeta o minimizante. Assim,

$$\tilde{\mathcal{L}}^{(t)} = \sum_{n=1}^{N} \left(g_n f_t(x_n) + \frac{1}{2} h_n f_t(x_n)^2 \right) + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2. \tag{2.58}$$

Definindo $I_j = \{n|q(x_n) = j\}$, i.e., os tuplos cujo mapeamento dado por q resultam na folha j, podemos reescrever a equação (2.58),

$$\tilde{\mathcal{L}}^{(t)} = \sum_{j=1}^{T} \left((\sum_{n \in I_j} g_n) w_j + \frac{1}{2} (\sum_{n \in I_j} h_n + \lambda) w_j^2 \right) + \gamma T. \tag{2.59}$$

Supondo uma estrutura q fixa e resolvendo a equação,

$$\frac{\mathrm{d}\tilde{\mathcal{L}}^{(t)}}{\mathrm{d}w_j} = \sum_{n\in I_j} g_n + w_j(\sum_{n\in I_j} h_n + \lambda) = 0, \tag{2.60}$$

obtemos o minimizante w_j^* relativo à folha j

$$w_j^* = -\frac{\sum_{n\in I_j} g_n}{\sum_{n\in I_j} h_n + \lambda}. \tag{2.61}$$

Substituindo na equação (2.59) obtemos o mínimo,

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{n=1}^{T} \frac{(\sum_{n\in I_j} g_n)^2}{\sum_{n\in I_j} h_n + \lambda} + \gamma T. \tag{2.62}$$

A equação (2.62) pode ser utilizada como score de uma determinada estrutura q de uma árvore. É impossível enumerar todas as estruturas possíveis para a estrutura q. Em vez disso, utilizamos um algoritmo *greedy* para encontrar as melhores divisões conhecido como *Exact Greedy Algorithm for Split Finding*, implementado no scikit-learn [17]. Sejam I_E e I_D os conjuntos de nós após feita a divisão à esquerda e direita, respetivamente, com I=$I_E \cup I_D$, então

$$\mathcal{L}_{split} = \frac{1}{2}\left(\frac{(\sum_{n\in I_E} g_n)^2}{\sum_{n\in I_E} h_n + \lambda} + \frac{(\sum_{n\in I_D} g_n)^2}{\sum_{n\in I_D} h_n + \lambda} - \frac{(\sum_{n\in I} g_n)^2}{\sum_{n\in I} h_n + \lambda}\right). \tag{2.63}$$

A equação (2.63) é utilizada para determinar as melhores divisões. Para além do termo regularizador da função objetivo $\mathcal{L}(\phi)$, ver equação (2.55), é comum atualizar os pesos através de um fator η (semelhante a um *learning rate*), deixando mais espaço para melhorias ao modelo por parte de futuras árvores e também subamostrar as variáveis/colunas com o intuito de evitar *overfitting*.

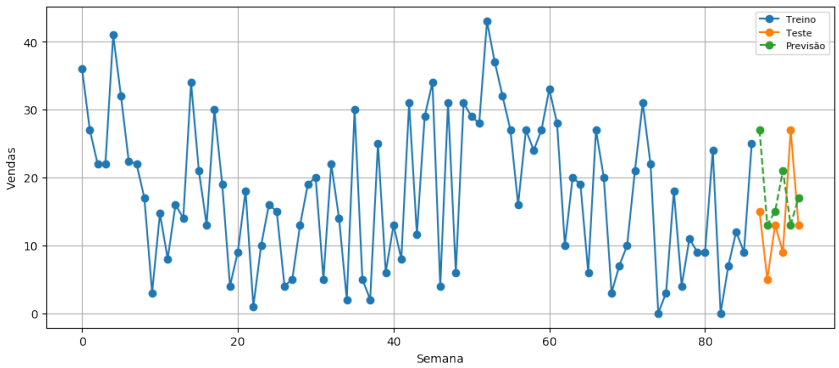

Figura 2.17: Previsão das vendas de uma peça de substituição para as próximas 6 semanas pelo método XGBoost.

2.4.2 SVR

Suponha-se $\{(x_n, t_n)\}_{n\in\{1,..,N\}}$ um *dataset*, com $t_n \in \mathbb{R}$ a variável dependente de $x_n \in \mathbb{R}^D$, que se pretende prever.. Para $x_n \in \mathbb{R}^D$ a previsão é dada por

$$y(x_n) = w^T \phi(x_n) + b. \tag{2.64}$$

Aqui ϕ é uma função que representa uma transformação que será melhor explanada adiante. Pretendemos minimizar a função objetivo

$$\min_{w\in\mathbb{R}} \left(C \sum_{n=1}^{N} E_\epsilon(y(x_n) - t_n) + \frac{1}{2}||w||^2 \right), \tag{2.65}$$

onde C é uma constante inversa de regularização e E_ϵ é a *insensitive error function* apresentada por Vapnik, em 1995, [46], pp.181-183, que é dada por

$$E_\epsilon(y(x_n) - t) = \begin{cases} 0 \qquad se \quad |y(x_n) - t| < \epsilon. \\ |y(x_n) - t| - \epsilon, \text{ caso contrário.} \end{cases}$$

Valores maiores de C fazem com que o modelo reduza o erro na fase de treino, à custa de maior tempo de treino e eventual menor capacidade de generalização do modelo. As condições dadas pelo ϵ-tubo obrigam a que $y_n - \epsilon \leq t_n \leq y_n + \epsilon$. A introdução de variáveis *slack* $\xi_n \geq 0$ e $\hat{\xi}_n \geq 0$ permitem relaxar estas condições para

$$t_n \leq y(x_n) + \epsilon + \xi_n \quad e \tag{2.66}$$

$$t_n \geq y(x_n) - \epsilon - \hat{\xi}_n. \tag{2.67}$$

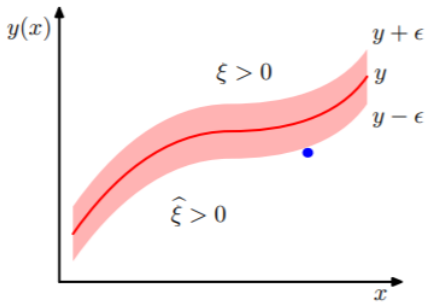

Figura 2.18: Função ϵ-tubo com váriaveis *slack*.
Fonte: [18], p.341.

Portanto, estamos perante um problema de otimização não linear com restrições

$$\min_{w\in\mathbb{R}}\left(C\sum_{n=1}^{N}(\xi_n+\hat{\xi}_n)+\frac{1}{2}||w||^2\right)$$

sujeito a

$$\begin{aligned}
&t_n \leq y(x_n)+\epsilon+\xi_n, \quad n\in\{1,...,N\},\\
&t_n \geq y(x_n)-\epsilon-\hat{\xi}_n, \quad n\in\{1,...,N\},\\
&\xi_n \geq 0, \quad n\in\{1,...,N\},\\
&\hat{\xi}_n \geq 0, \quad n\in\{1,...,N\}.
\end{aligned}$$

O minimizante é encontrado através das condições de Karush-Kuhn-Tucker (KKT), [19], pp.64-65. Considerando a função de Lagrange

$$\mathcal{L}(w,\hat{\xi},\xi,b)=\frac{1}{2}||w||^2+C\sum_{n=1}^{N}(\xi_n+\hat{\xi}_n)+\sum_{n=1}^{N}\alpha_n(t_n-y(x_n)-\epsilon-\xi_n)$$

$$+\sum_{n=1}^{N}\hat{\alpha}_n(y(x_n)-\epsilon-\hat{\xi_n}-t_n)-\sum_{i=1}^{N}(\lambda_n\xi_n+\hat{\lambda}_n\hat{\xi_n}).$$

Assim, pelas condições de KKT e atendendo a que $y(x_n)=w^T\phi(x_n)+b$, obtemos

$$\frac{\partial\mathcal{L}}{\partial w}=0 \implies w=\sum_{n=1}^{N}(\alpha_n-\hat{\alpha}_n)\phi(x_n), \tag{2.68}$$

$$\frac{\partial\mathcal{L}}{\partial b}=0 \implies \sum_{n=1}^{N}(\alpha_n-\hat{\alpha}_n)=0, \tag{2.69}$$

$$\frac{\partial\mathcal{L}}{\partial \xi_n}=0 \implies \lambda_n+\hat{\alpha}_n=C, \quad n\in\{1,...N\}, \tag{2.70}$$

$$\frac{\partial\mathcal{L}}{\partial \hat{\xi}_n}=0 \implies \hat{\lambda_n}+\alpha_n=C, \quad n\in\{1,...N\}. \tag{2.71}$$

As condições supra são conhecidas como as condições de estacionariedade. Seguem as chamadas condições de não negatividade

$$\alpha_n,\hat{\alpha}_n,\lambda_n,\hat{\lambda}_n\geq 0, \quad n\in\{1,...,N\}. \tag{2.72}$$

Das equações (2.70)-(2.72), seguem as relações

$$0 \leq \alpha_n \leq C, \quad n \in \{1, ..., N\}, \tag{2.73}$$
$$0 \leq \hat{\alpha}_n \leq C, \quad n \in \{1, ..., N\}, \tag{2.74}$$
$$0 \leq \lambda_n \leq C, \quad n \in \{1, ..., N\}, \tag{2.75}$$
$$0 \leq \hat{\lambda}_n \leq C, \quad n \in \{1, ..., N\}. \tag{2.76}$$

Finalmente, as condições de complementariedade são dadas por

$$\hat{\alpha}_n(t_n - y(x_n) - \epsilon - \xi_n) = 0 \quad n \in \{1, ..., N\}, \tag{2.77}$$
$$\alpha_n(y(x_n) - \epsilon - \hat{\xi}_n - t_n) = 0, \quad n \in \{1, ..., N\}, \tag{2.78}$$
$$-\lambda_n \xi_n = 0, \quad n \in \{1, ..., N\}, \tag{2.79}$$
$$-\hat{\lambda}_n \hat{\xi}_n = 0, \quad n \in \{1, ..., N\}. \tag{2.80}$$
$$\tag{2.81}$$

Substituinto a equação (2.68) na equação (2.64) concluímos

$$y(x) = \sum_{n=1}^{N} (\alpha_n - \hat{\alpha}_n)\phi(x_n)^T \phi(x) + b. \tag{2.82}$$

Os vetores de suporte são os x_n para os quais $\alpha_n \neq 0$ ou $\hat{\alpha}_n \neq 0$.
Note-se que, em geral, pode-se substituir $\phi(x_n)^T \phi(x)$ por $K(x_n, x)$, i.e., uma função núcleo (kernel) definida de acordo com o Teorema de Mercer, ver [url:14]. Esta função pretende mapear os dados para uma outra dimensão com recurso a produtos internos, onde seja mais fácil aplicar uma regressão linear.

Modelo	K(x,y)
Kernel linear	$(x^T y + c)$
Kernel polinomial	$(\alpha x^T y + c)^p$
Kernel RBF	$\exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$
Kernel sigmoide	$tanh(\alpha x^T y + c)$

Tabela 2.2: Funções de Kernel mais comuns.

Substituindo a função de kernel na equação (2.82) obtemos

$$y(x) = \sum_{n=1}^{N} (\alpha_n - \hat{\alpha}_n) K(x_n, x) + b \tag{2.83}$$

Definição 2.4.1 (Função coerciva)**.** *Uma função* $f : \mathbb{R}^n \to \mathbb{R}$ *diz-se coerciva, se*

$$\lim_{||x|| \to +\infty} f(x) = +\infty. \tag{2.84}$$

Uma vez que temos as condições:

1. $f(w, \xi, \hat{\xi}, C) = C \sum_{n=1}^{N} (\xi_n + \hat{\xi}_n) + \frac{1}{2}||w||^2$ é uma função convexa e coerciva;
2. As restrições são funções lineares, logo convexas.

O candidato a minimizante determinado pelas condições de KKT é na verdade o minimizante global do problema, ver Corolário 4.45, [19], p.77. O facto da função f ser coerciva, não interfere para efeitos de minimizante global, mas é importante para garantir a convergência dos métodos iterativos utilizados na prática.
Esta secção sobre o método SVR teve por base [18, 19, 20, 21, 22].

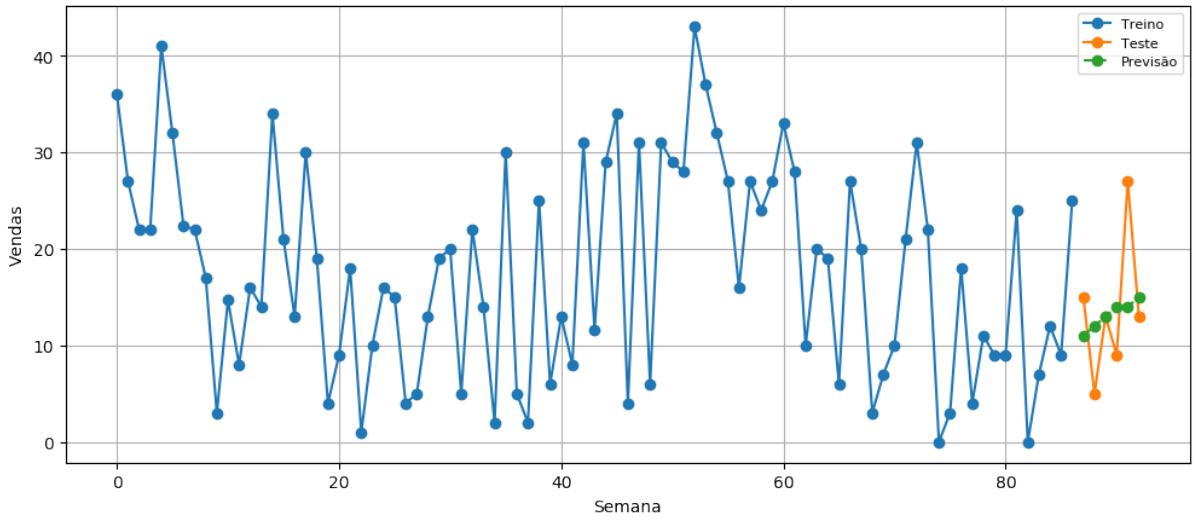

Figura 2.19: Previsão das vendas de uma peça de substituição para as próximas 6 semanas pelo método SVR.

2.5 Redes neuronais

As redes neuronais têm a sua primeira aparição em 1943, pela colobaração entre o neuroanatomista Warren McCulloch e o matemático Walter Pitts. Em 1958, Frank Rosenblatt cria o perceptrão [27] e, em 1960, Bernard Widrow e Ted Hoff, criam o *ADALINE* [28]. Nas décadas de 50 e 60, os investigadores julgavam, utopicamente, que as redes neuronais tinham capacidade ilimitada e podiam resolver qualquer problema, mas, em 1969, Minsky e Papert demonstraram que o perceptrão básico de Rosenblatt não resolvia problemas simples como o

XOR (ver [url:15]) e que os computadores da época não tinham capacidade de processamento suficiente para treinar os inúmeros pesos das redes neuronais com muitas camadas e neurónios [26]. O interesse pelas redes neuronais estagnou até ao final da década de 80. Com o crescimento da computação paralela e distribuída, aliado à forma eficiente de calcular os gradientes da função de custo através do algoritmo da retropropagação do erro, popularizado por Rumelhart, Hinton e Williams em 1986 [29], o interesse pelas redes neuronais cresceu, mas devido à concorrência com métodos mais simples como o SVR, que são determinísticos e garantem sempre minimizante global, o interesse prático generalizado nas redes neuronais só começou nos primeiros anos deste milénio. As aplicações das redes neuronais são vastas, desde o processamento de imagem até ao processamento de voz. Nesta secção, estamos interessados em utilizar redes neuronais para o nosso problema das séries temporais. Vamos começar pela arquitetura mais simples, denominado MLP, e por conceitos gerais das redes neuronais que são transversais à maior parte das arquiteturas/modelos. As redes neuronais também pertencem à classe de modelos de aprendizagem supervisionada, portanto aqui também é necessário transformar o problema num problema de aprendizagem supervisionada (cf. Figura 2.15).
À semelhança do que se fez para os métodos SVR e XGBoost, em redes neuronais supõe-se $\{(x_n, t_n)\}_{n\in\{1,..,N\}}$ um *dataset*, com $t_n \in \mathbb{R}$ a variável dependente de $x_n = (x_{n1}, x_{n2}, ..., x_{nD}) \in \mathbb{R}^D$, que se pretende prever.

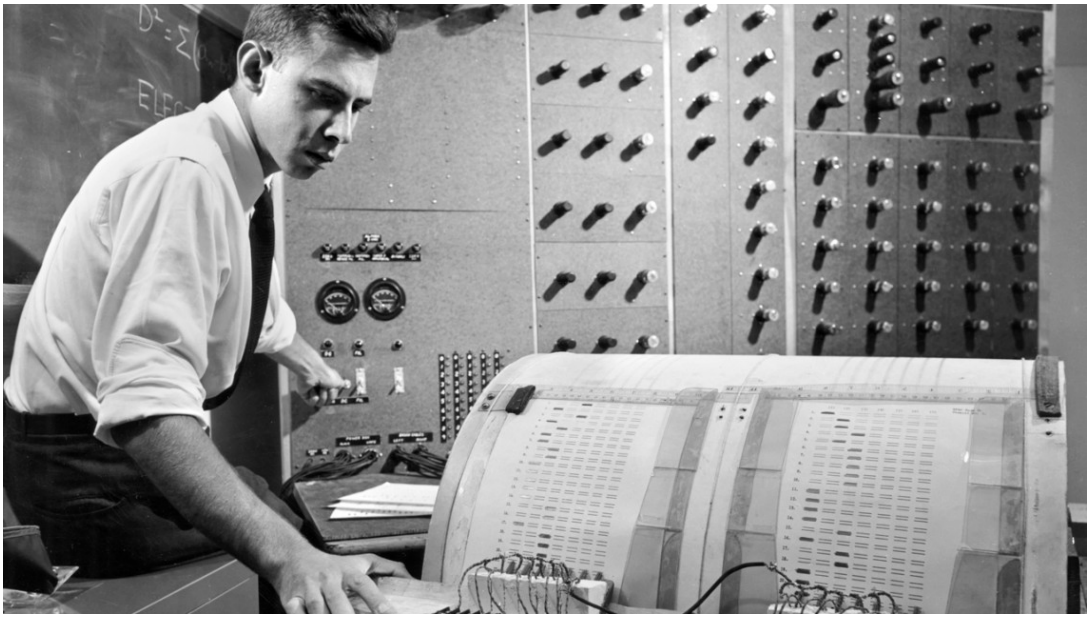

Figura 2.20: Frank Rosenblatt em volta do perceptrão.
Fonte:[url:16].

2.5.1 MLP

A arquitetura mais básica de redes neuronais é o MLP. A Figura 2.21 ilustra esta arquitetura e demonstra pictoricamente como podemos obter a previsão y_n a partir dos últimos D *lags*. O MLP é composto por camadas de input, escondidas e output e pretende aproximar a função que prevê o próximo valor da série temporal, dados os últimos D *lags*. Nesta arquitetura, todos os neurónios da camada i-1 estão ligados à camada i. $w_{i,j}^{(k)}$ representa o peso entre o neurónio j da camada k-1 e o neurónio i da camada k. Supondo que todas as n camadas escondidas têm M neurónios temos um total de $DM + (n-1)M^2 + M$ pesos para treinar, o que justifica o elevado esforço computacional necessário para treinar uma rede neuronal.

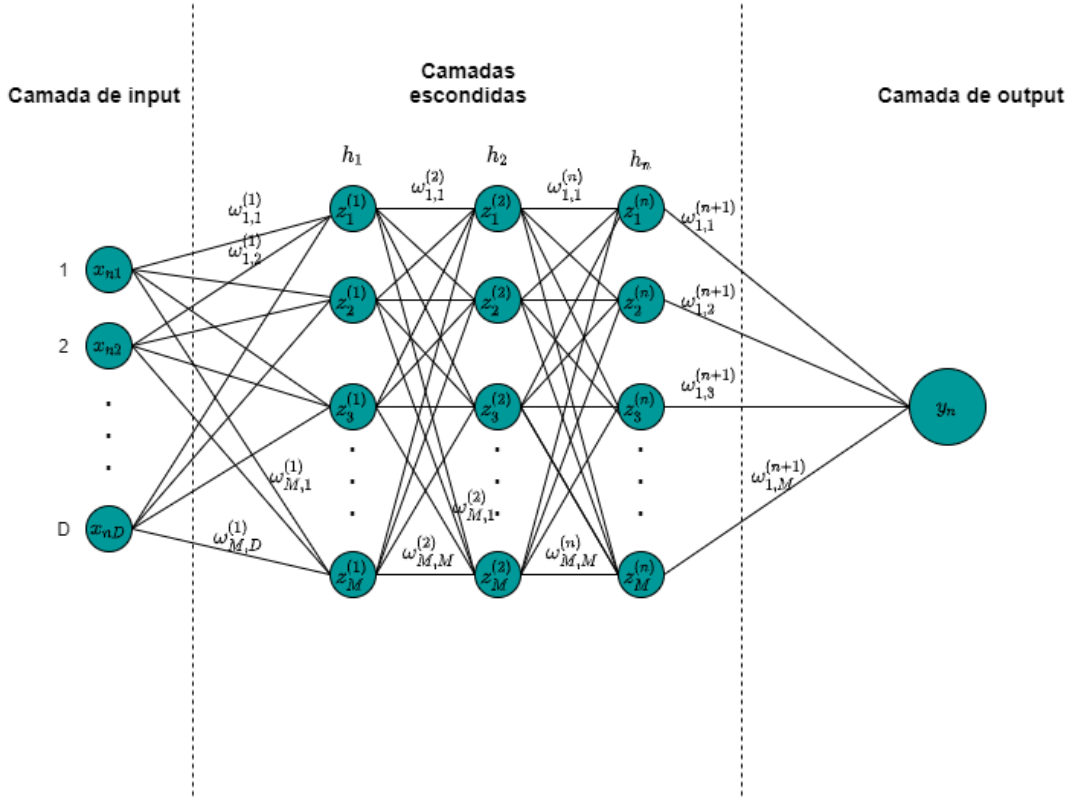

Figura 2.21: Representação gráfica de um MLP.

O valor dos neurónios da primeira camada escondida é calculado com recurso a combinações lineares do *input* da seguinte forma

$$a_j^{(1)} = \sum_{i=1}^{D} w_{j,i}^{(1)} x_{ni} + w_{j,0}^{(1)} \quad j \in \{1, ..., M\}, \tag{2.85}$$

onde M representa o número de neurónios da primeira camada escondida e $w_{j,0}^{(1)}$ é conhecido

como *bias*, que pode ser suprimido considerando o *input* artificial $x_{n0} = 1$. Desta forma,

$$a_j^{(1)} = \sum_{i=0}^{D} w_{j,i}^{(1)} x_{ni} \quad j \in \{1, ..., M\}. \tag{2.86}$$

Para $i \in \{1, ..., M\}$, cada neurónio $a_i^{(1)}$, da primeira camada escondida, tem uma função h_1 associada, diferenciável e não linear. Esta função h_1 é denominada de função de ativação e transforma os neurónios $\left\{a_i^{(1)}\right\}_{i\in\{1,...,M\}}$ por meio da fórmula

$$z_j^{(1)} = h_1(a_j^{(1)}) \quad j \in \{1, ..., M\}. \tag{2.87}$$

Noutras aplicações, nomeadamente em problemas de classificação, é costume utilizar também uma função de ativação na camada de *output*. No caso das séries temporais, tal é desprovido de sentido, utilizando-se a função identidade. Considerando que todas as camadas escondidas têm M neurónios, obtemos a nossa previsão através da fórmula

$$y_n = y(x_n, w) = \sum_{i=0}^{M} w_{1,i}^{(n)} z_i^{(n)}, \tag{2.88}$$

onde $\left\{z_j^{(n)}\right\}_{j\in\{1,...,M\}}$ são calculados recursivamente a partir dos neurónios da camada (n-1) por,

$$z_j^{(n)} = h_n\left(\sum_{i=0}^{M} w_{j,i}^{(n-1)} z_i^{(n-1)}\right), \quad j \in \{1, ..., M\}. \tag{2.89}$$

Treino de um MLP

Os pesos de uma rede neuronal são calculados à custa da minimização de uma função de custo diferenciável, que mede a diferença entre a previsão e o valor real. Um exemplo possível de uma função de custo é o somatório dos erros ao quadrado dado por

$$E(w) = \frac{1}{2}\sum_{n=1}^{N} (y_n - t_n)^2. \tag{2.90}$$

A determinação do minimizante da função de custo é, geralmente, obtido através de métodos iterativos, e.g., o método do gradiente descendente, [19], pp.27-35. Caso a atualização dos pesos seja feita no final de percorrer todo o conjunto de treino, diz-se que é feita em modo *batch*. A atualização em modo *batch* pelo método do gradiente descendente é dada por

$$w^{(k)} = w^{(k-1)} - \eta \nabla E(w^{(k-1)}) \quad \eta \in]0, 1], \tag{2.91}$$

onde η representa a taxa de aprendizagem. Alternativamente, se for possível escrever $E(w) =$

$\sum_{n=1}^{N} E_n(w)$, podemos atualizar os pesos pelo método do gradiente descendente em modo *online*, i.e., após passagem de cada tuplo por

$$w^{(k)} = w^{(k-1)} - \eta \nabla E_n(w^{(k-1)}) \quad \eta \in]0, 1]. \tag{2.92}$$

A atualização dos pesos pára quando $\nabla E(w^{(k-1)}) < \epsilon \approx 0$, se for feita em modo *batch* ou quando $\nabla E_n(w^{(k-1)}) < \epsilon \approx 0$, se for feita em modo *online*. Em sinergia com este critério de paragem ou de forma isolada pode ser acrescentado um limite máximo de iterações ou de *epochs*, sob pena da atualização nunca terminar em casos de impossibilidade de cumprimento do primeiro critério de paragem apresentado. Cada *epoch* representa uma passagem completa pelo *dataset* utilizado para treino por parte do método de otimização.
Conforme se constata pelas equações 2.91 (*batch*) e 2.92 (*online*), o método do gradiente descendente atualiza os pesos no sentido negativo do mesmo. A escolha adequada de η é primordial para efeitos de convergência do algoritmo, dado que um valor de η baixo, próximo de zero, requer um maior número de iterações até convergir, com elevada probabilidade do minimizante encontrado ser local. Por outro lado, um valor de η elevado, próximo de 1, pode inadvertidamente fazer com que o método divirja. Posto isto, não se recomenda utilizar um valor de η fixo, pois queremos um valor de η adaptativo que seja elevado inicialmente para permitir uma convergência rápida e que vá diminuindo à medida que estejamos próximo do mínimo para obter melhores aproximações do minimizante w, que se pretende que seja o minimizante global, embora na maior parte dos casos atinjamos minimizantes locais. A maioria dos métodos de otimização do momento já utilizam uma taxa de aprendizagem adaptativa. Para além do método do gradiente descendente básico, existem outros métodos, e.g., *Newton*, *quasi-Newton*, *conjugate gradients* [47]. Em 2015, Kingma e Ba [31] apresentaram um método de otimização denominado por *ADAM*, que se tem tornado dominante na maior parte das aplicações que utilizam redes neuronais. Este método combina as vantagens do *AdaGrad* [30] e *RMSProp* [48], p.7 . Embora os conceitos teóricos sobre a convergência do *ADAM* apenas sejam válidos sob a condição da função objetivo ser convexa que, em geral, não ocorre, os autores do *ADAM* verificaram que o método proposto apresentava melhores resultados do que os métodos mais populares da época, para um *dataset MNIST* de imagens, utilizando como arquitetura um MLP com 2 camadas escondidas, cada uma com 1000 neurónios (cf. Figura 2.22).

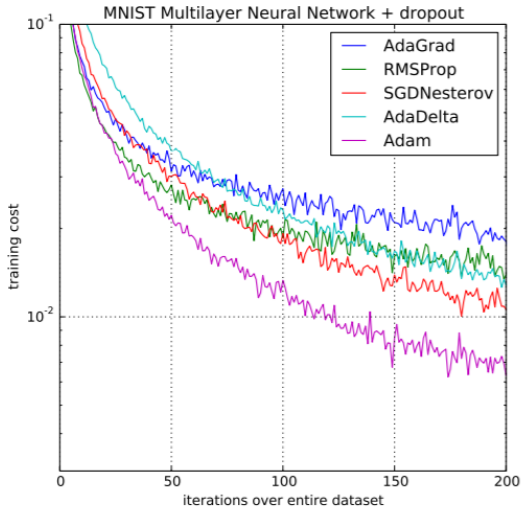

Figura 2.22: Comparação entre *ADAM* e outros métodos de otimização populares. Fonte: [31], p.7.

Retropropagação do erro

A atualização dos pesos através de um determinado método de otimização envolve o cálculo do gradiente. Devido aos vários pesos de uma rede neuronal e à necessidade de calcular derivadas parciais, a tarefa de calcular o gradiente pode tornar-se computacionalmente exigente. O algoritmo da retropropagação do erro permite mitigar esse problema, pois calcula o gradiente de forma astuciosa, com recurso à famosa regra da cadeia. Pal e Prakash [2], pp.66-68, proporcionam um exemplo de aplicação do algoritmo da retropropagação do erro, que é bastante claro e no qual nos baseamos.

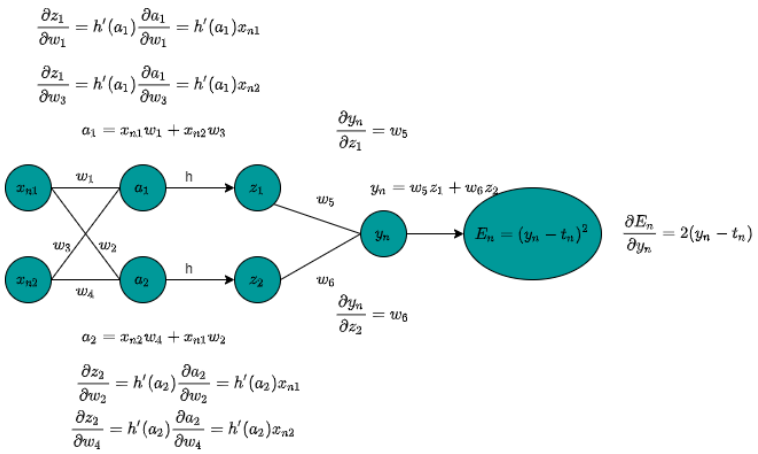

Figura 2.23: Retropropagação do erro num MLP

Aplicando a regra da cadeia à arquitetura da Figura 2.23, obtemos as derivadas

$$\frac{\partial E_n}{\partial w_1} = \frac{\partial E_n}{\partial y_n}\frac{\partial y_n}{\partial z_1}\frac{\partial z_1}{\partial w_1} = 2(y_n - t_n)w_5 + h'(a_1)x_1, \tag{2.93}$$

$$\frac{\partial E_n}{\partial w_2} = \frac{\partial E_n}{\partial y_n}\frac{\partial y_n}{\partial z_2}\frac{\partial z_2}{\partial w_2} = 2(y_n - t_n)w_6 + h'(a_2)x_1. \tag{2.94}$$

Este pequeno exemplo, ilustra o funcionamento deste algoritmo para calcular derivadas. O nome do algoritmo deve-se ao facto dos erros se propagarem no sentido inverso ao cálculo realizado no método MLP.

Funções de ativação

Se não existissem funções de ativação não lineares, o modelo MLP pós-treino seria linear, semelhante a um ARIMA. Utilizando funções de ativação não lineares, conseguimos que o nosso modelo se ajuste a problemas de previsão não lineares.

Ativação	f(x)
Tangente hiperbólica	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$
Sigmoide	$\frac{1}{1+e^{-x}}$
Rectified linear unit (ReLU)	$\max(x, 0)$

Tabela 2.3: Funções de ativação mais comuns.

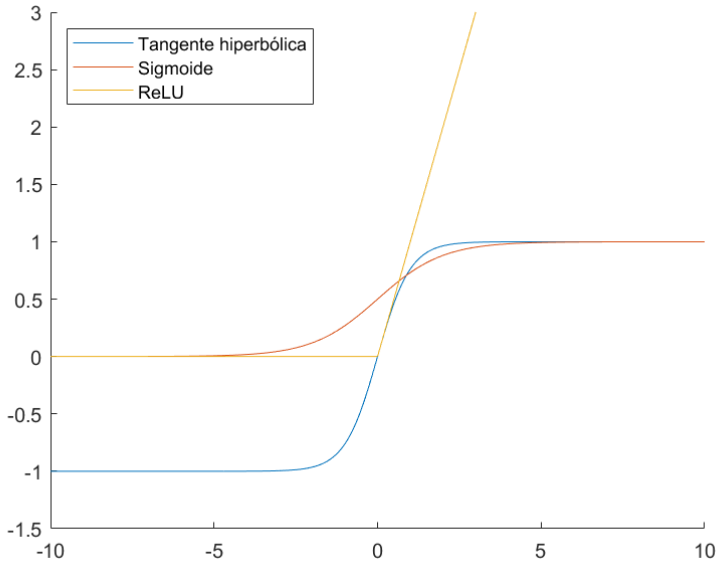

Figura 2.24: Funções de ativação mais comuns.

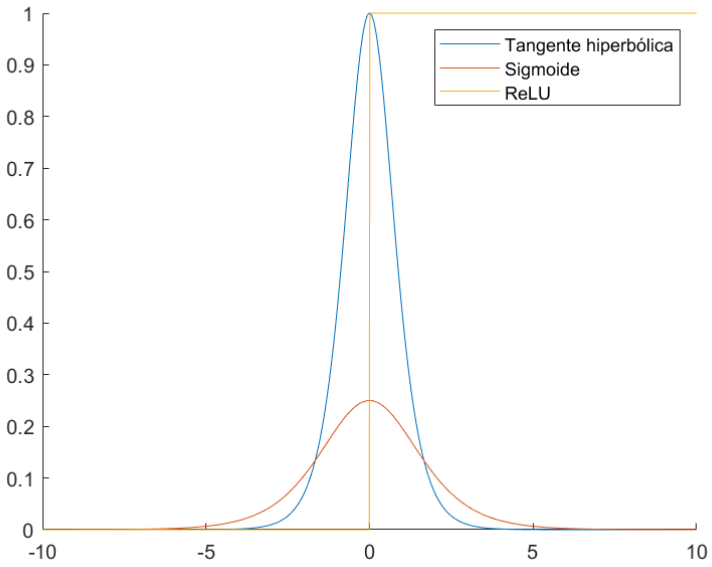

Figura 2.25: Primeira derivada das funções de ativação mais comuns.

A função de ativação mais utilizada em aplicações reais é a ReLU devido à sua simplicidade e pelo facto de evitar o problema da dissipação do gradiente, que consiste na multiplicação sucessiva de derivadas com valores inferiores a 1, aquando da utilização do algoritmo da retropropagação do erro, podendo terminar precocemente a atualização dos pesos pelo critério de paragem $\nabla E(w) < \epsilon$, embora não tenhamos atingido um mínimo local. Mesmo que se decida não utilizar este critério de paragem, utilizando apenas um número máximo de iterações ou *epochs*, com valores de gradiente próximos de zero, a atualização dos pesos dada pelas equações 2.91 (*batch*) e 2.92 (*online*) é pouco efetiva. Conforme podemos constatar pelo gráfico das derivadas, cf. Figura 2.25, as funções de ativação sigmoide e tangente hiperbólica sofrem do problema da dissipação do gradiente, dado que ao utilizar o algoritmo da retropropagação do erro, estamos a multiplicar constantemente valores inferiores a um, podendo resultar em $\nabla E(w) < \epsilon$. Devido à mudança abrupta, não desejável, em torno de 0 da função de ativação ReLU, em 2017, membros da equipa *Google Brain* propuseram uma nova função de ativação [32] , com forma semelhante à ReLU, batizada como *Swish*, dada por

$$f(x) = \frac{x}{1 + e^{-\beta x}}. \tag{2.95}$$

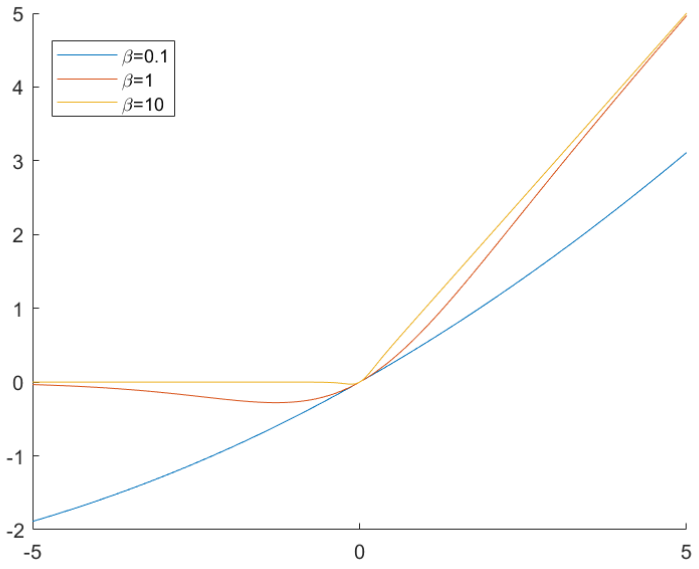

Figura 2.26: Função de ativação Swish para diferentes valores de β.

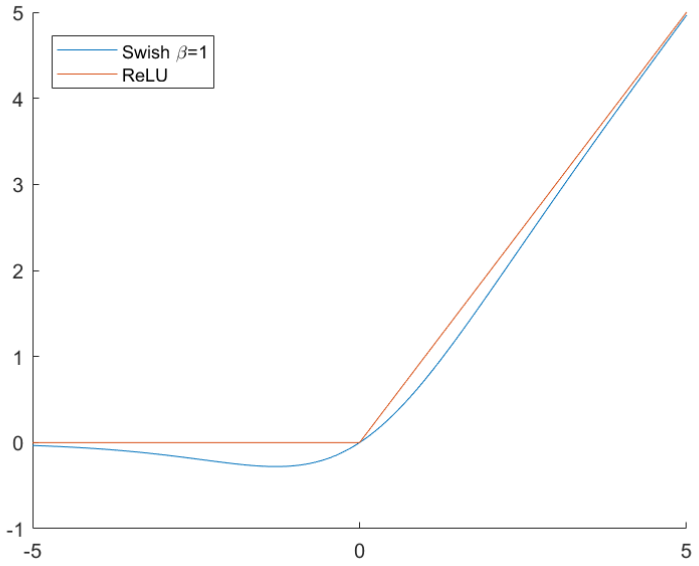

Figura 2.27: Comparação entre ReLU e Swish com $\beta = 1$.

Os autores da função de ativação *swish*, Prajit Ramachandran, Barret Zoph e Quoc V. Le, verificaram melhorias ao nível da eficácia da previsão dos modelos, pela simples troca da função de ativação ReLU pela *Swish*.

Regularização num MLP

A forma mais simples de fazer regularização e transversal a qualquer modelo é com recurso ao termo regularizador quadrático

$$\hat{E}(w) = E(w) + \lambda \frac{||w||^2}{2}, \quad \lambda \in \mathbb{R}.$$

Não obstante, esta técnica de regularização pode apresentar problemas ao nível de escala [18], pp.257-258.

Em aplicações práticas, a forma de regularização que tem ganho destaque no campo das redes neuronais é uma técnica que consiste na remoção aleatória, com probabilidade $p \in]0, 1[$ de alguns neurónios e respetivas ligações, cf. Figura 2.28. Esta técnica de regularização é conhecida como *dropout*.

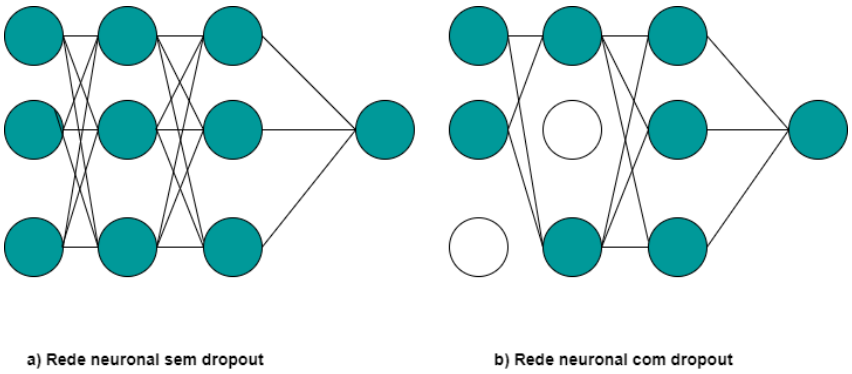

Figura 2.28: MLP *dropout*.

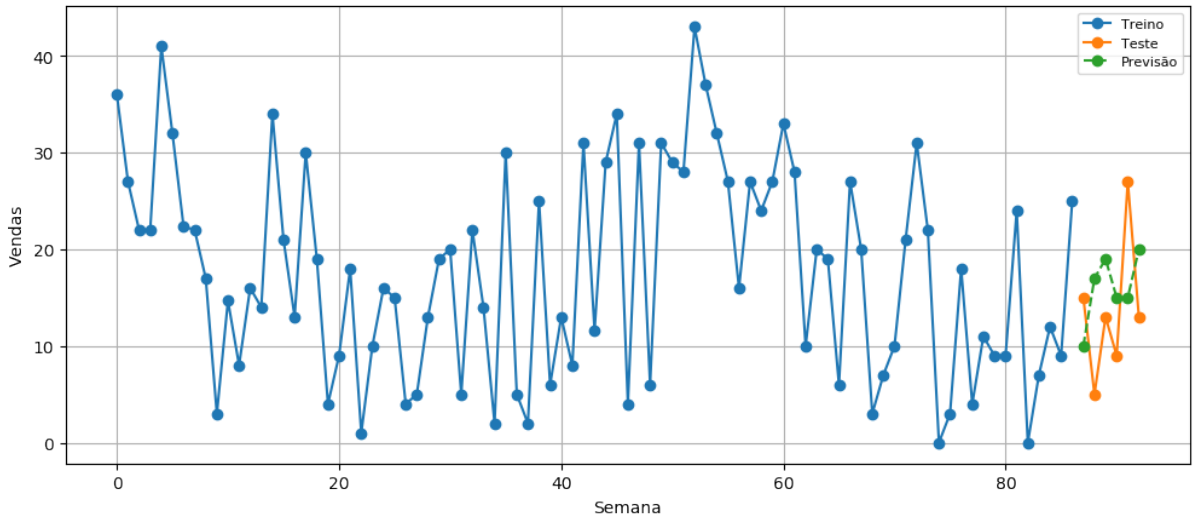

Figura 2.29: Previsão das vendas de uma peça de substituição para as próximas 6 semanas por um MLP.

2.5.2 CNN

A referência usual para as CNNs é o artigo de 1998/9 realizado por Le Cun et al.[49], apesar de Fukushima ter trabalhado em arquiteturas muito semelhantes, mas com outro nome em 1980 [51]. Krizhevsky et al. popularizaram esta arquitetura em 2012 com um trabalho realizado na área de processamento de imagem [50].
A CNN é uma classe de rede neuronal utilizada sobretudo para processamento de imagens, i.e., *datasets* com dados bidimensionais. As aplicações são variadíssimas, e.g., deteção de cancro e outras patologias em imagem médica, deteção e reconhecimento facial, sistemas de piloto automático, etc. A utilização das CNNs em detrimento do MLP para classificar imagens, deve-se ao facto das CNNs conseguirem utilizar toda a informação da imagem sem lidar com o embaraço de treinar centenas de milhões de pesos. Com efeito, tal redução no número de pesos é possível devido à forma astuta como as CNNs utilizam filtros para captar características em torno de determinada vizinhança. Apesar de geralmente associada a processamento de imagens, esta arquitetura também pode ser utilizada para construir modelos em séries temporais. A Figura 2.30 demonstra uma arquitetura com uma camada convolucional, com a qual podemos prever y_n, com base nos últimos D *lags*. O nome destas redes neuronais está relacionado com o facto de utilizarem o operador de convolução discreto

$$f_k = (x_n * w) = \sum_{i=1}^{K} x_{n(k-1+i)} w_i, \quad k \in \{1, ..., D-K+1\}, \tag{2.96}$$

onde $w = (w_1, ..., w_K) \in \mathbb{R}^K$ é conhecido como filtro de dimensão K e $\{f_k\}_{k\in\{1,...,D-K+1\}}$ são os elementos do *feature map*. A estes elementos do *feature map* é aplicada uma determinada função de ativação h, pelo que $f_k^* = h(f_k)$ para $k \in \{1, ..., D-K+1\}$. Os valores $\{f_k^*\}_{k\in\{1,...,D-K+1\}}$ seguem para uma camada *pooling* onde é aplicada a transformação dada por

$$p_j = \max(f_j^*, f_{j+1}^*, ..., f_{j+M-1}^*), \quad j \in \left\{1, 1+M, 1+2M, ..., \lfloor \frac{D-K+1}{M} \rfloor\right\} \quad (\textit{max pooling}), \tag{2.97}$$

$$p_j = \frac{f_j^* + f_{j+1}^* + ... + f_{j+M-1}^*}{M}, \quad j \in \left\{1, 1+M, 1+2M, ..., \lfloor \frac{D-K+1}{M} \rfloor\right\} \quad (\textit{mean pooling}), \tag{2.98}$$

onde M é o tamanho utilizado na camada *pooling*.

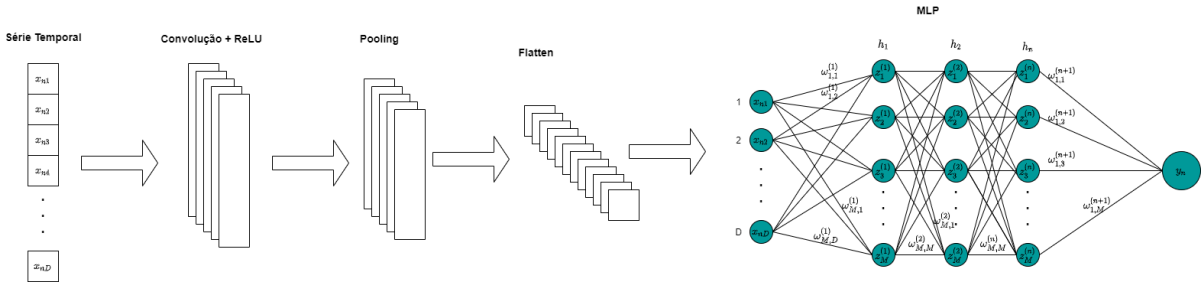

Figura 2.30: Arquitetura CNN com uma camada convolucional para séries temporais.

Atendendo à Figura 2.30, constata-se que, após o *input* passar pelas várias camadas, a previsão é dada através de um MLP. Para perceber melhor como estas redes funcionam, vamos recorrer a um exemplo ilustrativo.

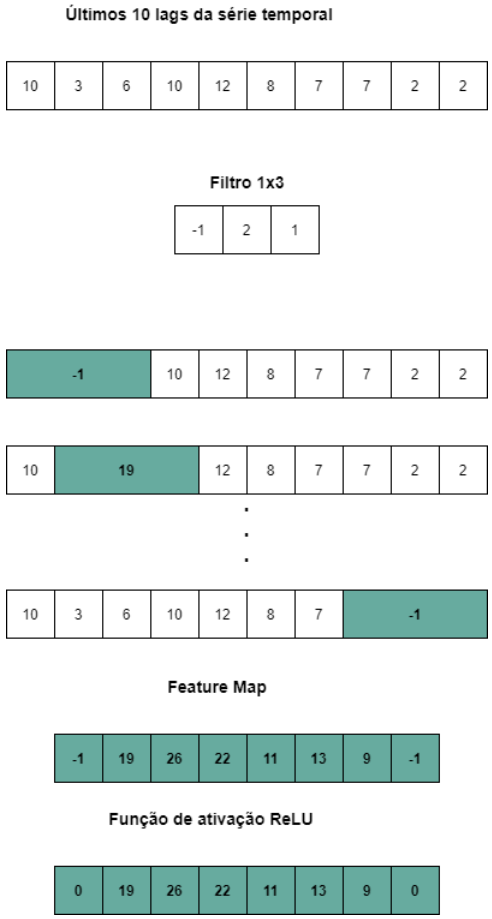

Figura 2.31: Filtro num CNN com aplicação da função de ativação ReLU.

Na prática, a operação de filtragem (cf. Figura 2.31 e equação 2.96) é aplicada vastas vezes, uma vez que filtros distintos captam características diferentes da série temporal, e.g., podemos utilizar dois filtros, um para captar a componente tendencial e outro a componente

sazonal. O *output* resultante da filtagrem segue para uma nova camada, *pooling*, na Figura 2.32 utilizou-se um *pooling* de tamanho 2 (podia ser doutro tamanho) e recorreu-se à função média, alternativamente também é costume utilizar a função máximo, que não é tão robusta para *outliers*.

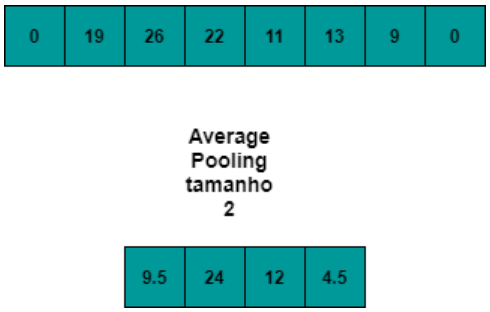

Figura 2.32: *Average Pooling* de tamanho 2 num CNN.

Deste nosso exemplo, os 10 *lags* da série temporal transforam-se num vetor de dimensão 4 após passagem pelas várias camadas, todavia, na prática utilizam-se vários filtros, supondo 5 filtros, após a operação de *flatten* (cf. Figura 2.30) teríamos 20 *inputs* para o nosso MLP, através dos quais se obteria a previsão y_n. A função de ativação sugerida no exemplo ilustrativo foi a ReLU pela sua simplicidade e boas propriedades, mas qualquer outra função de ativação não linear seria viável. Os coeficientes utilizados nos filtros não são fixos *a priori*, são pesos que são treinados e atualizados com recurso a um dos métodos de otimização discutidos anteriormente.

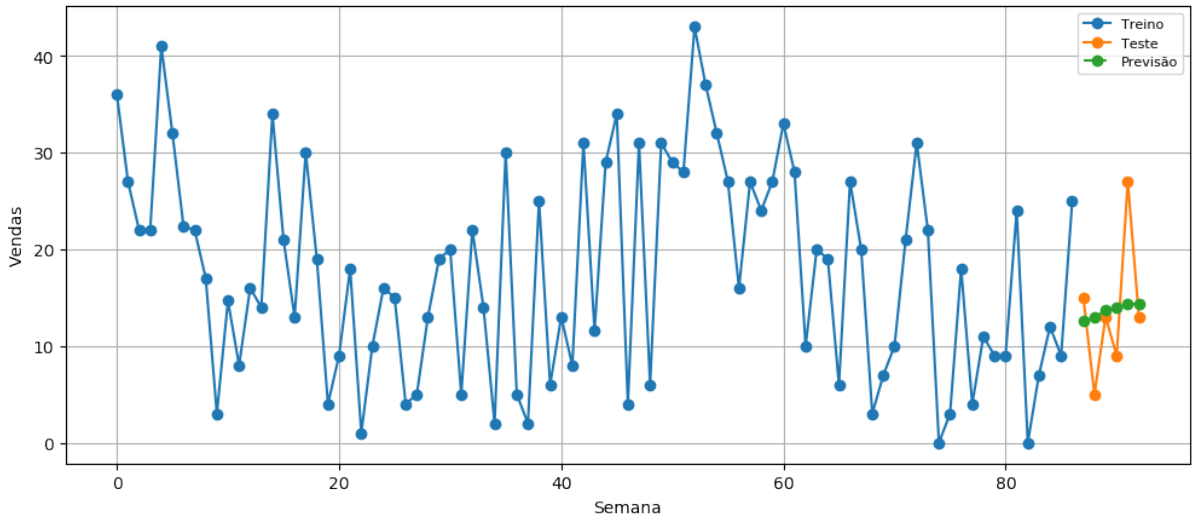

Figura 2.33: Previsão das vendas de uma peça de substituição para as próximas 6 semanas atrvés de uma .

2.5.3 LSTM

O MLP pretende prever y_n com base nos últimos D *lags*, no entanto, esta arquitetura parte do pressuposto que $x_{n1}, ..., x_{nD} \in \mathbb{R}$ são não correlacionados, o que não é verdade dada a natureza ordinal das séries temporais. Conforme vimos, as CNNs conseguem captar relações numa determinada vizinhança através de filtros, mas não são capazes de detetar relações entre observações a longo prazo. As LSTMs são redes especializadas para trabalhar com dados sequenciais, que conseguem lidar com relações entre observações a longo prazo. Antes de abordar as LSTMs vamos abordar as RNNs básicas, que serviram de inspiração.

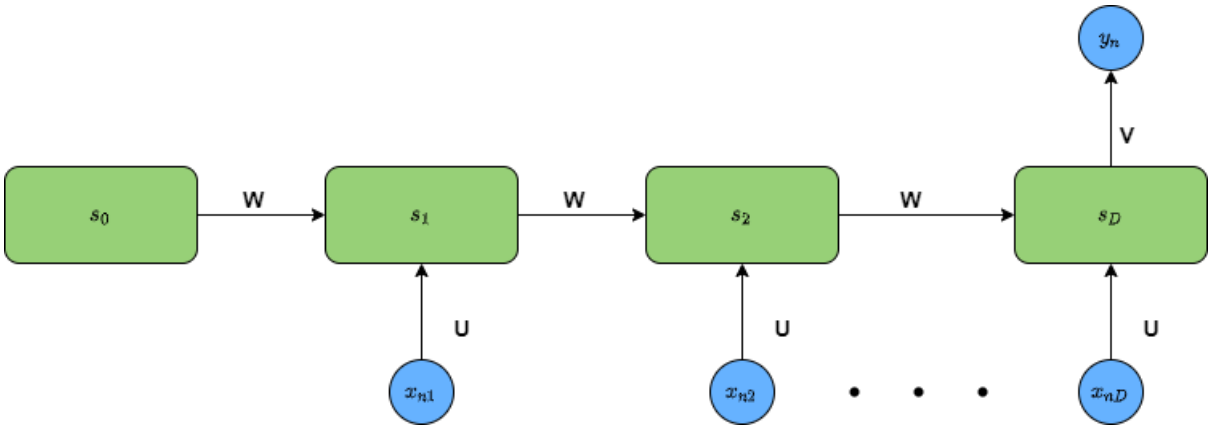

Figura 2.34: Arquitetura RNN *Many to One*

A Figura 2.34 demonstra a arquitetura RNN unidimensional. Nesta arquitetura, os pesos U,W e V são partilhados entre neurónios. Esta partilha de pesos confere duas vantagens:

1. Esforço computacional reduzido na fase de treino;

2. Possibilidade de integrar sequências de diferentes dimensões.

Não vamos tirar proveito da vantagem 2, embora em determinadas aplicações seja fulcral. Para $x_n = (x_{n1}, ..., x_{nD}) \in \mathbb{R}^D$ e $t \in \{1, ..., D\}$, cada neurónio $s_t \in \{s_1, ..., s_D\}$ recebe informação da observação x_{nt} associada e do neurónio s_{t-1}, cujo valor também tem influência de $x_{n(t-1)}$. É desta forma que as RNNs conseguem lidar com a natureza ordinal das séries temporais. As equações para cálculo do valor dos neurónios para um *timestep* t e previsão y_n são dadas por

$$s_t = h(Ux_{nt} + Ws_{t-1}), \tag{2.99}$$

$$y_n = Vs_D, \tag{2.100}$$

onde $U, W, V \in \mathbb{R}$. Note-se que, no caso de estarmos perante uma série temporal multivariada, antes da transformação para problema de aprendizagem supervisionada dado pela Figura 2.15, os pesos U, W, V seriam matrizes com dimensão adaptada à dimensão dos *inputs* que seriam vetores.

Os pesos, nesta arquitetura, são atualizados pelo algoritmo da retropropagação do erro pelo tempo, que é uma variante do algoritmo da retropropagação para esta arquitetura. Pelo facto dos pesos serem partilhados para todos os neurónios, ao calcular o gradiente, se $|W| < 1$ podemos estar perante o problema da dissipação do gradiente e se $|W| > 1$ temos o problema do gradiente explosivo, que pode conduzir a uma atualização exagerada dos pesos. Uma das técnicas possíveis para combater a dissipação do gradiente é utilizar a função de ativação ReLU, outra técnica consiste em inicializar os pesos com a matriz identidade. Todavia, nenhuma destas técnicas é suficientemente robusta, pelo que, em 1997, Sepp Hochreiter e Jürgen Schmidhuber propuseram o método LSTM [34].

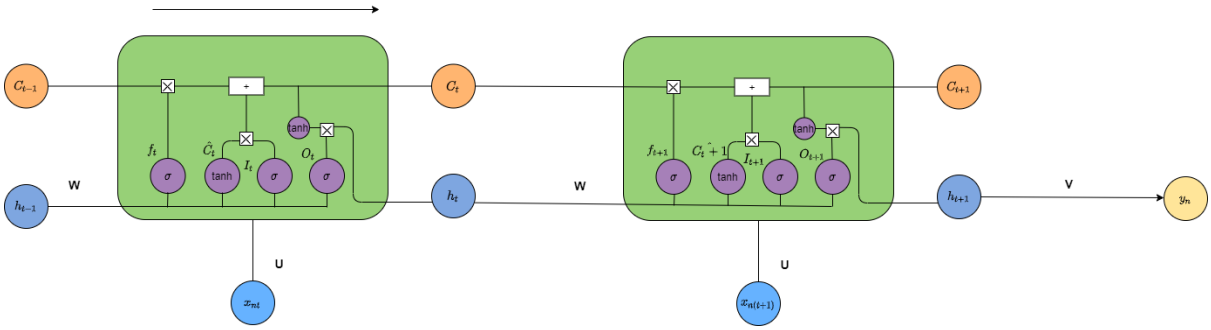

Figura 2.35: Arquitetura de um LSTM

Na Figura 2.35 o símbolo ⊠ representa a multiplicação elemento a elemento, que também pode ser denotado por ., i.e., o produto interno usual. As LSTMs, através de um mecanismo de portões, conseguem lidar melhor com o fluir da série temporal e captar dependências a longo termo, uma vez que têm:

- Portão de esquecimento f_t que através da função de ativação sigmoide elimina informação irrelevante;
- Portão de entrada I_t que mantém a informação relevante através da função ativação sigmoide;
- Portão de saída O_t que devolve uma versão filtrada do estado da célula também através da função de ativação sigmoide.

Os portões supra, são denominados de portões internos, mas existem outros portões, h_t e C_t, que representam o estado da célula corrente. Os portões h_t e C_t são atualizados em cada célula à custa da informação relativa à célula anterior, conjuntamente com o *timestep* atual x_{nt}.

Assim, cada célula recebe como *input* x_{nt}, h_{t-1} e C_{t-1} devolvendo para a próxima célula h_t e C_t. As equações de atualização de uma LSTM para um *timestep* t são usualmente[3]

$$f_t = \sigma(x_{nt}U_f + h_{t-1}W_f), \tag{2.101}$$

$$\hat{C}_t = tanh(x_{nt}U_c + h_{t-1}W_c), \tag{2.102}$$

$$I_t = \sigma(x_{nt}U_i + h_{t-1}W_i), \tag{2.103}$$

$$O_t = \sigma(x_{nt}U_o + h_{t-1}W_o), \tag{2.104}$$

$$C_t = f_t C_{t-1} + I_t \,.\, \hat{C}_t, \tag{2.105}$$

$$h_t = O_t \,.\, tanh(C_t), \tag{2.106}$$

$$y_n = V h_{t+1}. \tag{2.107}$$

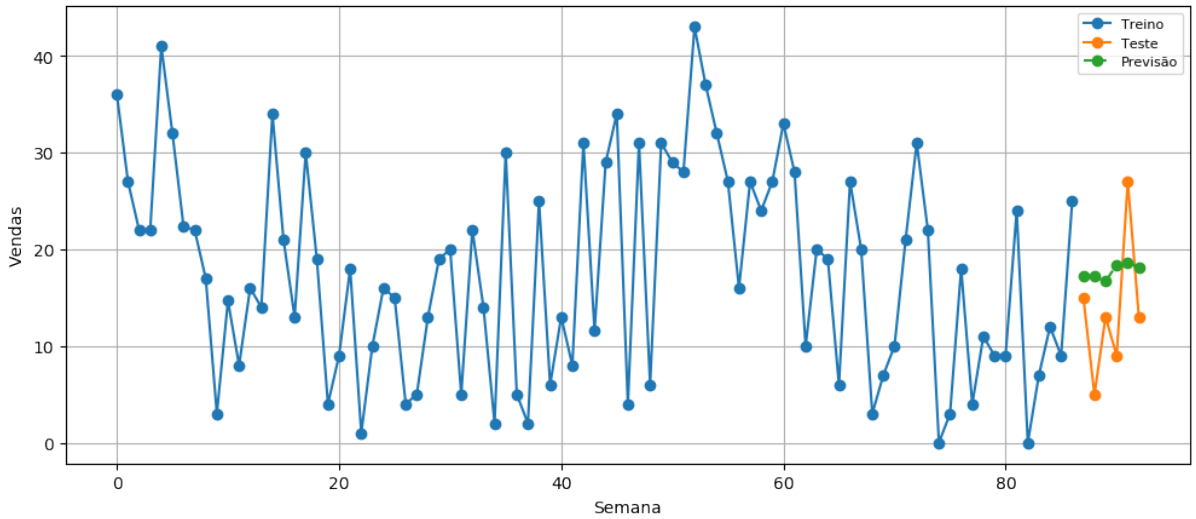

Figura 2.36: Previsão das vendas de uma peça de substituição para as próximas 6 semanas através do método LSTM.

2.6 Métodos de *Ensemble Learning-Stacking*

Neste capítulo foram apresentados 7 modelos diferentes (ARIMA, Holt-Winters, XGBoost, SVR, MLP, CNN e LSTM). Conforme se pode verificar pelas gráficos das previsões dos vários métodos que fomos mostrando ao longo do capítulo, as previsões são ainda inferiores ao desejado. Para melhorar a eficácia das previsões é possível criar um novo modelo que combine as características destes 7 modelos. Sejam $y_n^1, y_n^2, ..., y_n^7$ as previsões de cada um dos 7 modelos, podemos recorrer a um determinado modelo, por exemplo, MLP para determinar

[3]Embora na figura 2.35 se tenha utilizado W e U por simplicicação, W e U são diferentes, em função dos respetivos portões.

$\hat{y_n} = F(y_n^1, y_n^2, ..., y_n^7)$, i.e., a previsão do nosso modelo *ensemble*, que é calculado treinando neste caso um MLP sobre o *dataset* de treino $\{(y_n^i, t_n)\}_{i\in\{1,2,..,7\}}$. Esta abordagem, apesar de garantir em geral melhor eficácia, apresenta duas desvantagens:

- Esforço computacional elevado, devido à necessidade de treinar todos os 7 modelos no *dataset* original;
- Elevada probabilidade de *overfitting* dado o número reduzido de tuplos, igual ao número de modelos a utilizar, 7 neste caso.

2.7 Critérios de seleção do melhor modelo

Após treinar cada modelo na série temporal em estudo, é necessário escolher qual dos modelos produz melhores previsões. Essa seleção é feita calculando o erro entre as previsões e as observações de teste, i.e., que não foram utilizadas para treino. Os critérios mais populares para medir o erro das previsões são o Root mean squared error (RMSE) e Mean squared error (MSE), que são dados por

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(t_n - y_n)^2}, \tag{2.108}$$

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(t_n - y_n)^2. \tag{2.109}$$

Apesar de muito utilizados, o MSE e RMSE não conferem uma medida de erro relativo, não sendo possível interpretar se o erro é grande ou pequeno. Desta forma, é também utilizado o Mean absolute percentage error (MAPE) dado por

$$MAPE = \frac{100}{N}\sum_{n=1}^{N}\left|\frac{t_n - y_n}{t_n}\right|. \tag{2.110}$$

Apesar de ser um bom critério, não é possível calcular o MAPE quando $t_n = 0$ para algum $n \in \{1, ..., N\}$, pelo que nestes casos recomendamos utilizar o coeficiente de determinação, denotado por R^2 e dado por

$$R^2 = 1 - \frac{\sum_{n=1}^{N}(t_n - y_n)^2}{\sum_{n=1}^{N}(t_n - \mu)^2}, \tag{2.111}$$

onde μ é a média da série temporal. Temos que $R^2 \in]-\infty, 1]$, sendo que $R^2 = 1$ no caso do modelo ter comportamento perfeito na fase de teste, i.e., não existe erro. Por outro lado, um valor $R^2 < 0$ significa que o modelo se comporta pior do que considerando como previsão uma linha horizontal dada pelo valor médio μ. Posto isto, pretendemos sempre $R^2 > 0$, sendo que o valor de R^2 a partir do qual se considera o modelo aceitável depende da aplicação.

Capítulo 3

Infraestrutura, ETL, Pipeline

Até agora escrutinámos a intuição matématica subjacente a cada modelo utilizado neste projeto, essencial para estarmos conscientes das capacidades e limitações de cada método. Todavia, esta dissertação não tem apenas pretensões teóricas, pois necessitamos de um sistema informático eficiente e escalável que suporte as características do problema. Neste capítulo vamos apresentar:

1. As tecnologias adotadas neste projeto, bem como os motivos que justicam tais escolhas;
2. Um esquema com o fluxo de dados do projeto;
3. A escolha da linguagem de progamação para criar os modelos de previsão e livrarias utilizadas.

3.1 Docker

Um dos grandes problemas aquando da implementação de um sistema informático, prende-se com o facto da pouca portabilidade da solução devido a: sistemas operativos distintos, versões de *software* distintas, versões distintas nas linguagens de progamação e respetivas livrarias, etc. O docker, utilizado neste projeto, permite correr a nossa aplicação de forma isolada, com todas as dependências instaladas, em *containers*. Esta noção sugere que um *container* é semelhante a uma máquina virtual, o que é verdade, mas existem várias vantagens na utilização de *docker containers* que passamos a elencar:

- Todos os *containers* partilham o sistema operativo hospedeiro;
- Um *container* é mais veloz no arranque do que uma máquina virtual;
- mais leve.

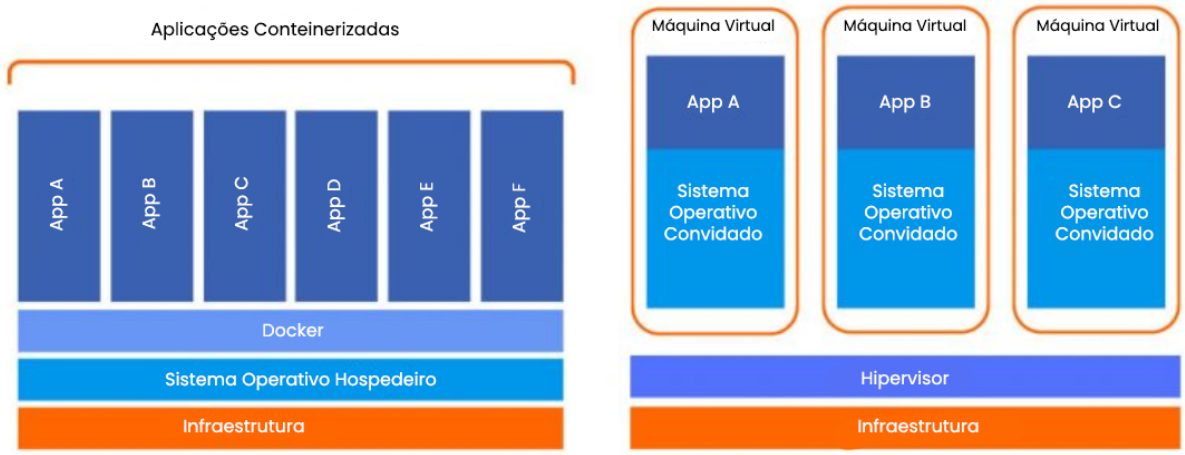

Figura 3.1: Comparação entre Docker e máquina virtual.
Fonte:[url:18].

Um *container* é uma instância executável de uma imagem criada por um *docker file*. A Figura 3.2 demonstra este ciclo. Um *docker file* pode ser escrito na linguagem yaml ou json, embora seja mais comum utilizar-se yaml, que é um superconjunto do json.

Figura 3.2: Criação de um *docker container* através de um *docker file*

As imagens *docker* são habitualmente gravadas para o *Docker Hub*, um repositório de imagens. Desta forma, qualquer progamador pode baixar imagens desenvolvidas pela sua equipa ou por outros progamadores.
Neste projeto, utilizou-se o ecossistema sbroETL que é uma imagem *docker* criada pelo Prof. Doutor Eugénio Rocha especializada para projetos de ciência de dados.

Imagem	Docker Hub
mariadb	[url:29]
adminer	[url:30]
metabase	[url:31]
directus	[url:32]

Tabela 3.1: Imagens do *Docker Hub* que integram o ecossistema sbroETL.

O mariadb é um sistema de gerenciamento de base de dados relacional dos criadores do MySQL, já o adminer permite criar uma *interface* gráfica para poder ver os registos das tabelas e poder fazer comandos SQL, compatível com vários sistemas de basos de dados (MySQL, PostgreSQL, MongoDB, Oracle, etc). Por fim, o metabase e directus são ferramentas de análise de dados e *dashboarding* que operam sobre a base de dados, permitindo criar gráficos de modo que a informação possa ser compartilhada com pessoas de várias áreas.

3.2 Pipeline do projeto

O Extract, transform, load (ETL) de um projeto diz respeito à fase de carregamento e tratamento de dados de múltiplas fontes para um destino singular. As fontes podem ser variadas, ficheiros excel, tabelas de bases de dados, ou no pior dos casos, dados não estruturados (ver [url:28]), e.g., vídeos e imagens. Em muitos projetos, a parte do ETL chega a ser mais complexa que a própria construção de modelos. Para o leitor interessado em saber mais sobre este assunto, recomendamos a obra de Jensen et al. [53]. Neste projeto o ETL é muito simples, conforme podemos ver pela Figura 3.4. Os dados do ficheiro vendas.csv com os campos: referência do material, quantidade, entrado em e classe da peça de substituição, antes de serem carregados para a tabela data_filtered, sofrem as seguintes transformações:

1. Agregação semanal das vendas;

2. Deteção de picos pelo algoritmo de Palshikar modificado;

3. O excesso de cada pico é repartido de forma uniforme pelas duas semanas anteriores;

4. Os passos 2 e 3 são respetidos K vezes, definido pelo progama.

Conforme mencionado no ponto 3, o tratamento de picos não é feito via imputação por um determinado modelo, mas distribuindo o excesso de cada pico pelas duas semanas anteriores de forma uniforme, isto porque, segundo a Bosch, esses picos devem-se a atrasos na notificação ao sistema, pelo que transportando algumas destas vendas para as semanas prévias, estamos a aproximarmo-nos do comportamento real da série temporal. De acordo com o indicado no ponto 4, pode ser favorável repetir este processo, dado que existem casos em que o pico é de tal forma acentuado, que ao repartir o excesso pelas duas semanas anteriores, estas tornam-se picos. Este tratamento de dados é fundamental, caso contrário estariamos a utilizar observações que não seguem o padrão da série temporal e que ao serem utilizadas para treino dos modelos poderiam comprometer a eficácia dos mesmos.

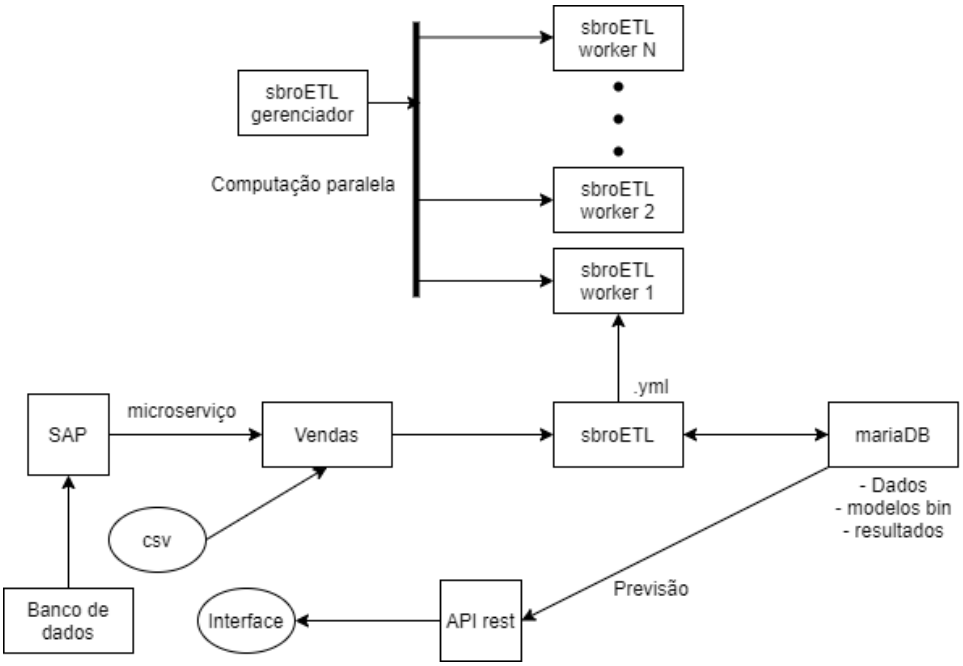

Figura 3.3: Modelo conceptual do pipeline do projeto.

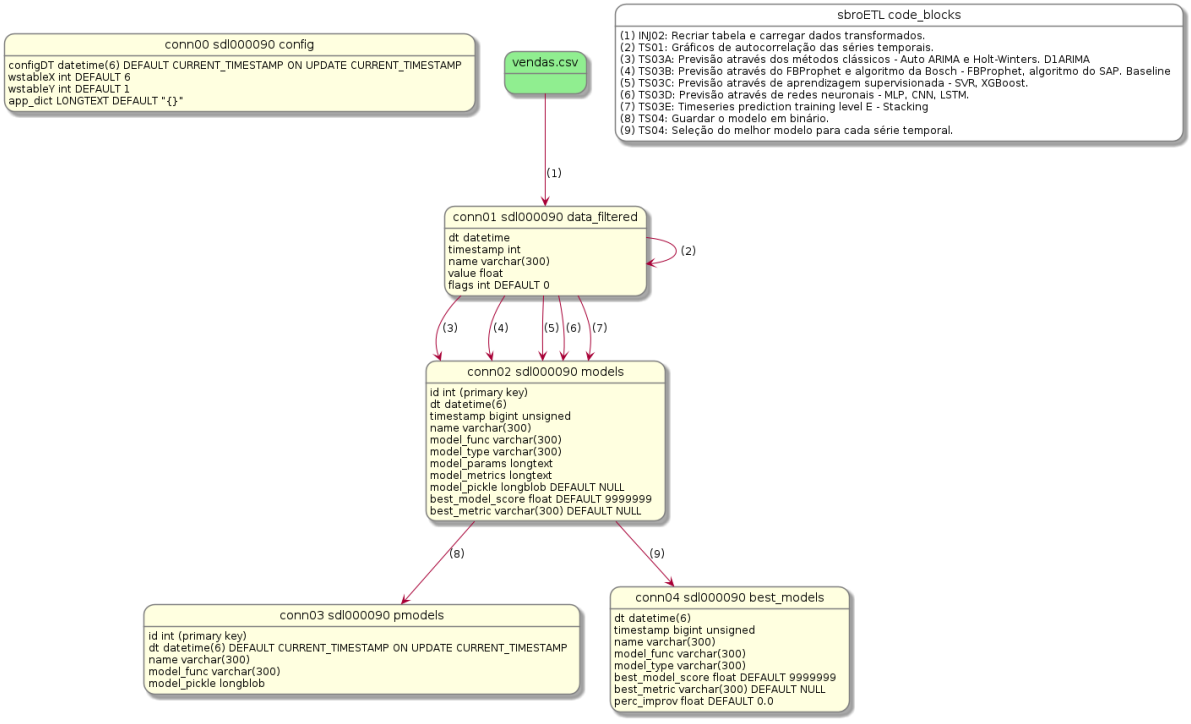

Figura 3.4: Modelo relacional e tabelas utilizadas no mariaDB.

3.3 Python para séries temporais

A linguagem utilizada para conceber os modelos foi o Python, uma vez que é *open source*, tem sintaxe simples, tem boa integração com outras tecnologias, permite progamação modular e podemos ainda tirar partido de livrarias especializadas na área de estatística e *machine learning*.

Modelo	Livraria	Documentação
ARIMA	pmdarima	[url:19]
Holt-Winters	statsmodels	[url:20]
FBProphet	prophet	[url:21]
SVR	sklearn	[url:22]
XGBoost	xgboost	[url:23]
MLP	keras	[url:24]
CNN	keras	[url:24]
LSTM	keras	[url:24]

Tabela 3.2: Modelos utilizados e respetivas livrarias Python.

A livraria pmdarima permite que o treino do modelo ARIMA seja feito de forma automática, i,e, a ordem de diferenciação d é determinada através de um teste de raízes unitárias (Kwiatkowski–Phillips–Schmidt–Shin (KPSS), Dickey-Fuller ou Phillips-Perron), permite escolher um intervalo para os parâmetros p e q, treinando modelos para todas as combinações dadas, escolhendo o melhor modelo e respetivos parâmetros p e q por minimização do AIC ou BIC. A livraria pmdarima permite ainda lidar com modelos SARIMA com acréscimo dos parâmetros P, D e Q, no entanto o espaço de parâmetros a pesquisar torna-se enorme, pelo que o treino torna-se lento em computadores pessoais. Uma vez que a grande maioria das séries temporais têm valores de venda nulos e a livraria statsmodel não permite séries temporais com valores nulos para o modelo multiplicativo do Holt-Winters, apenas se utilizou o modelo aditivo. Para os modelos de aprendizagem supervisionada e redes neuronais é preciso transformar a nossa série temporal num problema de aprendizagem supervisionada, o que nos obriga a escolher previamente o número de *lags* com os quais queremos efetuar a previsão, relembrar Figura 2.15. Após vários testes em pequenas amostras, concluiu-se que utilizar 3 *lags* seria o ideal, pois minimizava o erro em fase de teste. Os próprios gráficos das FACPs corroboram esta escolha, dado que para muitas séries temporais sugerem precisamente 3 *lags* significativos.

Capítulo 4

Resultados

4.1 Métodos univariados

Nesta secção, vamos apresentar os resultados obtidos para as primeiras 100 peças de substituição de classe A, apesar do nosso *dataset* ser composto por milhares de peças de substituição, entre as quais, 1659 são desta classe. O facto de estarmos apenas a apresentar resultados para 100 peças de substituição, justifica-se pela utilização de um computador pessoal, incapaz de reproduzir os resultados em tempo útil, com as seguintes características:

- Processador: Intel Core i5-8265U 1.6 GHz (6MB cache, até 3.9 GHz, 4 núcleos);
- Memória RAM: 8GB LPDDR3;
- Disco: PCIe Gen3 x2 NVMe 512GB M.2 SSD;
- Placa gráfica: Integrated Intel UHD Graphics 620 e NVIDIA GeForce MX250.

Após feita a agregação semanal, o *dataset* é composto pelas vendas semanais de 2018 até 2020 de cada peça de substituição, perfazendo um total de 157 semanas para cada peça de substituição. As tabelas que se seguem demonstram os resultados obtidos, fazendo variar a altura do ano em que se fazem as previsões de forma a estarmos sensíveis a potenciais diferenças na qualidade da previsão. O espaço de busca utilizado em treino para os hiperparâmetros dos modelos pode ser consultado em apêndice. A métrica utilizada foi o rmse6040, uma ligeira alteração ao RMSE, pois atribuiu-se ao erro um peso de 60% quando a previsão é por defeito e 40% quando a previsão excede o valor real. Esta ligeira alteração ao RMSE demonstra maior sensibilidade perante o contexto do problema, dado que previsões por excesso são menos gravosas do que previsões por defeito. A equação 4.1 demonstra de que forma é apurada a percentagem de melhoria relativamente ao método da Bosch, sendo que no caso do modelo vitorioso ser o método utilizado pela Bosch não é aplicável. Note-se que, caso a métrica escolhida

tenha valor zero para o método utilizado pela Bosch, então este é vitorioso não sendo aplicável o cálculo de percentagem de melhoria.

$$\text{Melhoria} = 100 \times \frac{\text{métrica método utilizado pela Bosch} - \text{métrica modelo vitorioso}}{\text{métrica método utilizado pela Bosch}} \quad (\%). \tag{4.1}$$

Modelo	#Vitórias	Tempo de execução	Melhoria média
ARIMA	6	2 minutos e 25 segundos	48.42%
Método da Bosch	13	0 segundos	N/A
FBProphet	11	2 minutos e 26 segundos	46.01%
Holt-Winters	8	5 segundos	45.71%
SVR	0	18 minutos e 39 segundos	N/A
CNN	4	7 minutos e 46 segundos	73.56%
LSTM	2	20 minutos e 21 segundos	61.01%
MLP	5	8 minutos e 3 segundos	68.22%
Stacking	48	155 minutos e 56 segundos	62.83%
XGBoost	3	36 segundos	61.54%
Melhoria média geral de 58.85% relativamente ao método da Bosch.			

Tabela 4.1: Previsão das semanas 108, 109 e 110 com *Stacking*.

Modelo	#Vitórias	Tempo de execução	Melhoria média
ARIMA	8	2 minutos e 21 segundos	45.06%
Método da Bosch	20	0 segundos	N/A
FBProphet	23	2 minutos e 24 segundos	38.52%
Holt-Winters	14	5 segundos	39.46%
CNN	7	7 minutos e 49 segundos	62.21%
LSTM	11	19 minutos e 38 segundos	63.98%
MLP	7	8 minutos e 1 segundos	44.13%
SVR	5	18 minutos e 23 segundos	60.03%
XGBoost	5	31 segundos	42.71%
Melhoria média geral de 47.01% relativamente ao método da Bosch.			

Tabela 4.2: Previsão das semanas 108, 109 e 110 sem *Stacking*.

Modelo	#Vitórias	Tempo de execução	Melhoria média
ARIMA	11	2 minutos e 19 segundos	69.11%
Método da Bosch	17	0 segundos	N/A
FBProphet	22	2 minutos e 16 segundos	45.37%
Holt-Winters	3	4 segundos	47.95%
CNN	10	minutos e 29 segundos	68.99%
LSTM	9	28 minutos e 40 segundos	72.91%
MLP	7	10 minutos e 42 segundos	58.05%
SVR	15	21 minutos e 24 segundos	54.28%
XGBoost	6	31 segundos	53.26%
Melhoria média geral de 57.69% relativamente ao método da Bosch.			

Tabela 4.3: Previsão das semanas 121, 122 e 123 sem *Stacking*.

Modelo	#Vitórias	Tempo de execução	Melhoria média
ARIMA	16	2 minutos e 26 segundos	31.70%
Método da Bosch	16	0 segundos	N/A
FBProphet	14	2 minutos e 25 segundos	35.03%
Holt-Winters	4	4 segundos	47.78%
CNN	18	10 minutos e 35 segundos	66.41%
LSTM	7	29 minutos e 27 segundos	55.31%
MLP	6	11 minutos e 21 segundos	57.51%
SVR	6	27 minutos e 6 segundos	39.94%
XGBoost	13	33 segundos	48.64%
Melhoria média geral de 47.44% relativamente ao método da Bosch.			

Tabela 4.4: Previsão das semanas 134, 135 e 136 sem *Stacking*.

Modelo	#Vitórias	Tempo de execução	Melhoria média
ARIMA	11	2 minutos e 48 segundos	33.19%
Método da Bosch	16	0 segundos	N/A
FBProphet	18	2 minutos e 24 segundos	39.68%
Holt-Winters	10	4 segundos	50.90%
CNN	8	13 minutos e 0 segundos	67.52%
LSTM	6	30 minutos e 32 segundos	55.98%
MLP	8	10 minutos e 58 segundos	66.71%
SVR	16	47 minutos e 2 segundos	56.06%
XGBoost	7	57 segundos	48.16%
Melhoria média geral de 50.38% relativamente ao método da Bosch.			

Tabela 4.5: Previsão das semanas 147, 148 e 149 sem *Stacking*.

Conclui-se que o modelo que apresenta melhores resultados no compromisso entre esforço computacional e eficácia na previsão é o FBProphet, no entanto não existe um modelo que domine de forma esmagadora, justificando-se a utilização de todos os modelos. Conforme esperado, quando utilizado, o *Stacking* venceu em 48 de 100 peças de substituição possíveis e melhorou os resultados em mais de 10%, todavia a probabilidade deste modelo sofrer de *overfitting* é bastante elevada, pelo que os resultados devem ser interpretados com cautela. Experimentámos ainda o SARIMA, cujos resultados estão presentes na Tabela 4.6 que, comparando com a tabela homóloga (Tabela 4.3), verifica-se um aumento na melhoria média geral de 7,23%. Apesar desta melhoria, o SARIMA levou mais de 107 minutos a correr do que o ARIMA.

Modelo	#Vitórias	Tempo de execução	Melhoria média
SARIMA	17	109 minutos e 38 segundos	78.56%
Método da Bosch	10	0 segundos	N/A
FBProphet	23	2 minutos e 18 segundos	70.64%
Holt-Winters	7	4 segundos	47.64%
CNN	7	10 minutos e 33 segundos	73.76%
LSTM	9	28 minutos e 33 segundos	81.19%
MLP	10	10 minutos e 50 segundos	39.15%
SVR	10	21 minutos e 20 segundos	59.31%
XGBoost	7	30 segundos	45.31%
Melhoria média geral de 64.92% relativamente ao método da Bosch.			

Tabela 4.6: Previsão das semanas 121, 122 e 123 com SARIMA e sem *Stacking*.

4.2 Métodos multivariados

Devido à existência de várias peças de substituição adstritas a equipamentos de termotecnologia crê-se que, enriquecedendo o nosso *dataset* com dados do clima, podemos diminuir o erro das nossas previsões. Relembrando o nosso *dataset*, dispomos de dados para 1659 peças de substituição de classe A, com vendas semanais desde 2018 até 2020. Para enriquecer o nosso *dataset* utilizámos a Application programming interface (API) *Visual Crossing Weather* (ver [url:17]). As variáveis utilizadas para prever o valor das vendas semanais são: temperatura mínima, temperatura máxima, temperatura média, humidade relativa, velocidade do vento, direção do vento, precipitação, profundidade da neve, visibilidade, nebulosidade e pressão atmosférica ao nível do mar. Através da API obtivemos os valores semanais predominantes das variáveis mencionadas acima para o território nacional entre 2018 e 2020. De seguida, apresentam-se alguns gráficos dessas variáveis climáticas onde se pode constatar uma sazonalidade bem vincada para a temperatura média.

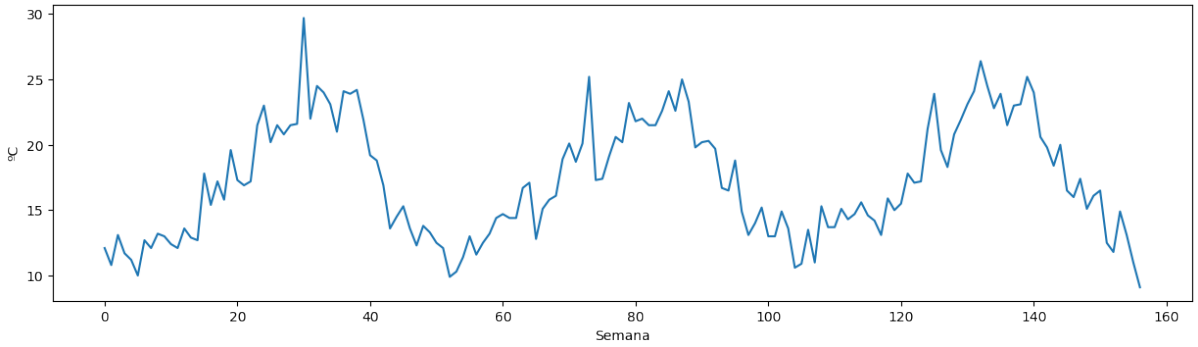

Figura 4.1: Temperatura média semanal em Portugal entre 2018 e 2020.

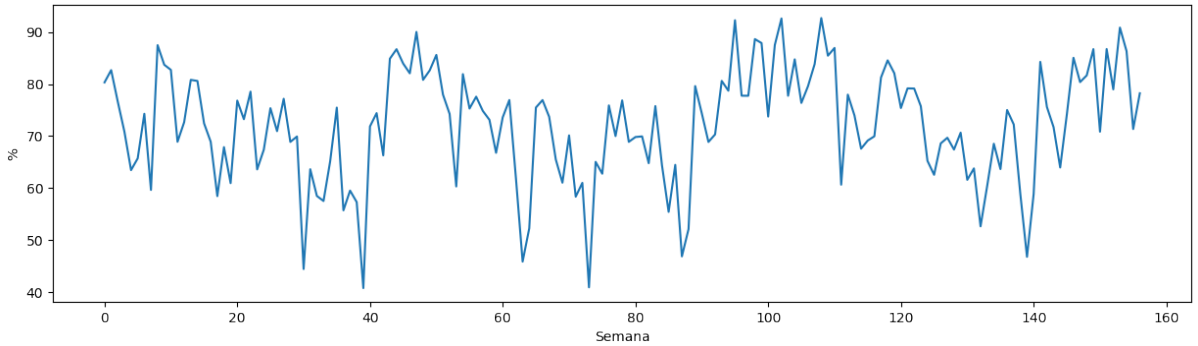

Figura 4.2: Humidade relativa semanal em Portugal entre 2018 e 2020.

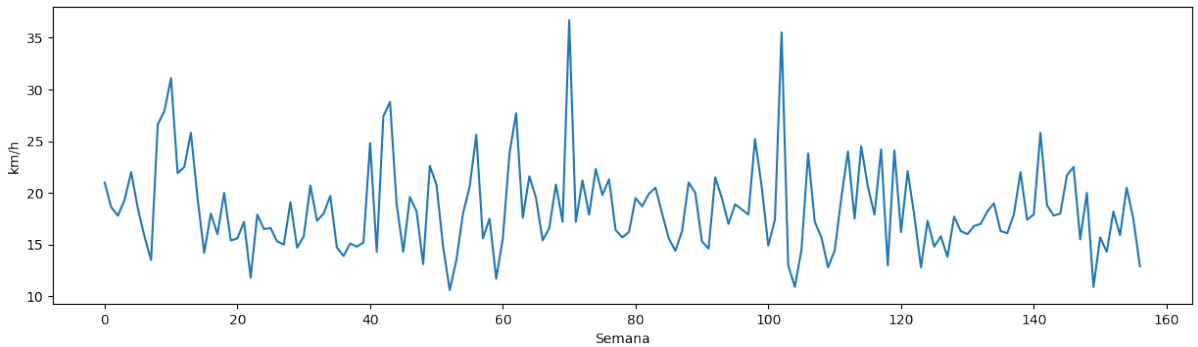

Figura 4.3: Velocidade do vento semanal em Portugal entre 2018 e 2020.

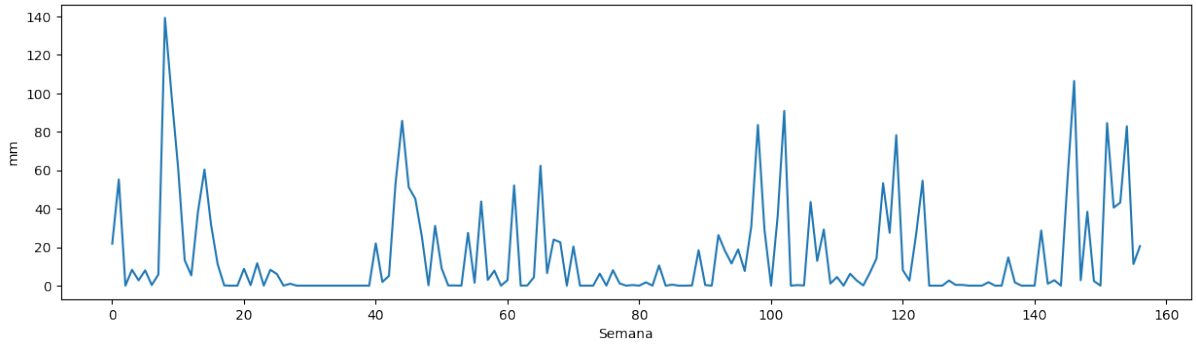

Figura 4.4: Precipitação semanal em Portugal entre 2018 e 2020.

Ao contrário do esperado, as variáveis do clima não permitem melhorar as previsões feitas na secção anterior, pelo contrário, as previsões pioram. A Figura 4.5 demonstra que praticamente não existe correlação entre as principais variáveis do clima e o valor das vendas semanais para as 1659 peças de substituição de classe A.

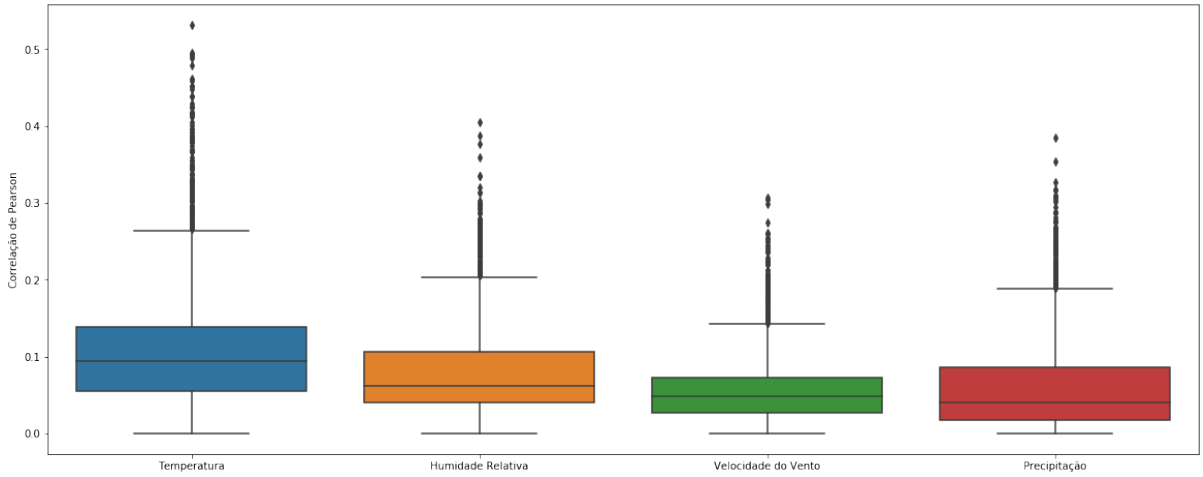

Figura 4.5: *Boxplot* para o valor absoluto das correlações de Pearson entre as variáveis do clima e vendas semanais das peças de substituição.

Em todo o caso, os valores próximos de zero para a correlação de Pearson entre as principais variáveis do clima e o valor das vendas não é conclusivo, dado que o coeficiente de correlação de Pearson verifica apenas a existência ou não de uma relação linear. Naturalmente que pode não existir uma relação linear, mas existir uma relação não linear. Para verificar que efetivamente os métodos univariados superam os multivariados e que não existe uma relação não linear vincada entre os dados do clima e o valor das vendas, utilizámos um modelo não linear sofisticado, LSTM, comparando-se o RMSE entre o LSTM univariado e LSTM multivariado (cf. figura 4.6). Claramente o LSTM univariado produz melhores previsões.

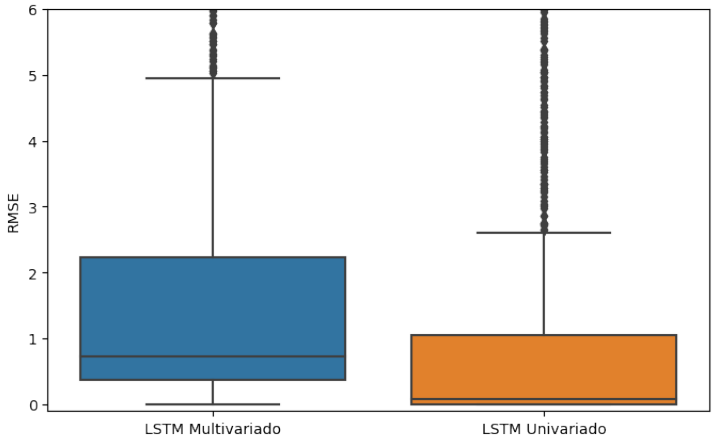

Figura 4.6: Comparação entre o LSTM multivariado e LSTM univariado.

Capítulo 5

Conclusão

O objetivo primário ao qual este projeto se propôs foi alcançado, na medida em que os modelos aqui propostos apresentam melhorias médias totais sempre superiores a 47% relativamente ao método utilizado pela Bosch para vários períodos do ano de 2020, conforme se pode verificar no capítulo dos resultados. Note-se que, o ano de 2020 foi um ano atípico para os mercados, em virtude da pandemia de COVID-19 provocada pelo novo coronavírus, SARS-CoV-2, ainda assim os modelos propostos apresentaram-se robustos ao fazer previsões neste ano, semana 105 a 157. Quanto ao compromisso entre esforço computacional e eficácia na previsão, os melhores modelos foram o FBProphet e ARIMA, no entanto ao nível da eficácia na previsão não seria possível obter estes resultados sem considerar os métodos de aprendizagem supervisionada, nomeadamente a utilização de redes neuronais que contabilizaram várias vitórias com valores de melhoria média muito elevados.
Conforme se percebeu pela secção referente aos resultados dos métodos multivariados, a nossa intuição falhou, já que, as variáveis explanatórias do clima não permitiram explicar o número de vendas. Efetivamente, as previsões foram piores do que as conseguidas pelos métodos univariados. Em virtude da ineficiência dos métodos multivariados e atendendo que, a utilização de uma só variável desfasada no tempo para auto-previsão, não absorve toda a informação necessária para prever o número de vendas. Consideramos que, para melhorar os modelos já obtidos, necessitamos de *datasets* que incorporem variáveis relevantes. A informação relativa a promoções feitas em determinada peça de substituição poderia ser integrada nos dados, bem como a informação referente a que tipo de peça de substituição se trata, se de um produto primário que ainda está na fase de produção em série, ou de um produto primário descontinuado com obrigação contratual de peças de substituição próxima do limite de garantia de 15 anos, ver Figura 1.1. Naturalmente que especialistas na área da gestão de *stock* e planeamento da produção detêm o *know-how* necessário para saber que fatores influenciam o número de vendas, podendo sugerir novas váriaveis para enriquecer o *dataset*.

Não tão importante quanto o enriquecimento do *dataset* com novas variáveis, mas ainda assim com alguma importância, reconhecemos que outros modelos não abordados neste trabalho poderiam garantir melhores resultados, e.g., modelos híbridos ou modelos baseados em distribuições de probabilidade. Para melhorar as previsões dos modelos com base em redes neuronais, poder-se-ia aumentar o espaço de busca dos hiperparâmetros e também correr o mesmo modelo múltiplas vezes, seguido de um *ensemble* para obter modelos mais consistentes, dada a natureza estocástica destes modelos. No entanto, o revés destas ideias seria o aumento desmesurado do esforço computacional em fase de treino, quer por adição de novos modelos no que concerne à tipologia, quer por necessidade de construção de mais modelos dentro da mesma tipologia.

Pessoalmente, este trabalho foi ao encontro das minhas expectativas, pois dificilmente existiria outro projeto que se ajustasse tanto às aplicações matemáticas do meu interesse. Durante a Licenciatura e Mestrado recebi a formação necessária para que agora estivesse apto para este projeto, no entanto nunca tinha desenvolvido um projeto em ciência de dados com todos os passos, i.e., desde o tratamento de dados em bruto, passando pela construção de modelos e acabando com um *dashboard*. Este trabalho permitiu desenvolver competências na área de séries temporais que no arranque para este trabalho eram praticamente nulas, aprofundar o conhecimento das linguagens Python e SQL, ter um primeiro contacto com ferramentas de *deploy*, como o *docker*, e ainda melhorar capacidades não técnicas como a autonomia, comunicação e trabalho em equipa. Encaro este trabalho como um início de um percurso na área emergente de ciência de dados que, devido ao seu ecleticismo, exige muita formação até se atingir um nível adequado de mestria.

Lista de Endereços Web

[url:1] https://www.bosch.pt/noticias-e-historias/1886-1905-da-primeira-oficina-a-fabrica/
[url:2] https://ec.europa.eu/growth/sectors/automotive_en
[url:3] https://www.bosch.pt/noticias-e-historias/1906-1925-globalizacao-e-um-novo-comeco/
[url:4] https://www.bosch.pt/noticias-e-historias/1946-1959-um-novo-comeco-e-reconstrucao/
[url:5] https://www.bosch.pt/a-nossa-empresa/a-nossa-historia/
[url:6] https://github.com/jbrownlee/Datasets/blob/master/airline-passengers.csv
[url:7] http://web.vu.lt/mif/a.buteikis/wp-content/uploads/2019/02/03_Tasks.html
[url:8] https://felzek.github.io/covid-project/Time_Series_Example_Image.jpg
[url:9] https://www.researchgate.net/profile/Pablo_Hernandez-Leal/publication/330227565/figure/fig2/AS:74688
[url:10] https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation
[url:11] https://miro.medium.com/max/1464/1*5vky1z29e1iO6iOvCTBJxg.png
[url:12] https://en.wikipedia.org/wiki/Gradient_boosting#Gradient_tree_boosting
[url:13] https://en.wikipedia.org/wiki/Ensemble_learning
[url:14] https://en.wikipedia.org/wiki/Mercer%27s_theorem
[url:15] https://en.wikipedia.org/wiki/XOR_gate
[url:16] https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon
[url:17] https://www.visualcrossing.com/weather-data
[url:18] https://vertigo.com.br/containers-vs-maquinas-virtuais/
[url:19] https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html
[url:20] https://www.statsmodels.org/dev/generated/statsmodels.tsa.holtwinters.ExponentialSmoothing.html
[url:21] https://facebook.github.io/prophet/docs/quick_start.html#python-api
[url:22] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html
[url:23] https://xgboost.readthedocs.io/en/latest/python/python_api.html
[url:24] https://keras.io/api/
[url:25] https://en.wikipedia.org/wiki/Bootstrapping_(statistics)
[url:26] https://en.wikipedia.org/wiki/Akaike_information_criterion
[url:27] https://en.wikipedia.org/wiki/Bayesian_information_criterion
[url:28] https://www.mongodb.com/unstructured-data
[url:29] https://hub.docker.com/_/mariadb
[url:30] https://hub.docker.com/_/adminer
[url:31] https://hub.docker.com/r/metabase/metabase
[url:32] https://hub.docker.com/r/directus/directus
[url:33] https://pt.wikipedia.org/wiki/Princ%C3%ADpio_de_Pareto
[url:34] https://pt.wikipedia.org/wiki/Curva_de_Lorenz

Bibliografia

[1] Brockell, P.J.; Davis, R.A. (1996). *Introduction to Time Series and Forecasting.* Springer-Verlag, New-York.

[2] Pal, Avishek; Prakash, PKS. (2017). *Practical Time Series Analisys.* Packt. Birmingham.

[3] Brownlee, Jason. (2018). *Deep Learning for Time Series Forecasting.* Machine Learning Mastery.

[4] McClellan, James H.; Schafer, Ronald W; Yoder, Mark A.. (2003). *Signal Processing First: A Multimedia Approach*, Prentice Hall.

[5] Braei, Mohammad; Wagner, Sebastian. (2020). Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art.

[6] Ferreira, Pedro. (2013). *Estimação e Selecção de Caos Determinístico em Séries Temporais Financeiras.*

[7] Bartlett, M. (1946). On the theoretical specification and sampling properties of autocorrelated time series. *Supplement to the Journal of the Royal Statistical Society*, 8(1), 27-41.

[8] Box, G.; Pierce, D.A. (1970). Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Models. *Journal of the American Statistical Association*, 65, 1509-1526.

[9] Ljung, G.; Box, G. (1978). On a measure of lack of fit in time series models. Biometrika, 65, 297-303.

[10] Fuller, W.A. (1976). *Introduction to statistical time series.* New York: John Wiley and Sons. ISBN 0-471-28715-6.

[11] Chen, T.; Guestrin, C.. (2016). XGBoost: A Scalable Tree Boosting System. *In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD x27;16. New York, NY, USA: ACM, pp. 785–794.

[12] Phillips, P. C. B.; Perron, P. (1988). Testing for a unit root in time series regression. *Biometrika* 75 335-346.

[13] Kwiatkowski, D.; Phillips, P.; Schmidt, P.; Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54, 159-178.

[14] Dickey, D.A.; Fuller, W.A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association* 74 427-431

[15] Evans, G. B. A.; Savin, N. E. (1981). Testing for unit roots: 1. *Econometrica* 49 753-779.

[16] Palshikar, Girish. (2009). Simple Algorithms for Peak Detection in Time-Series.

[17] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; M. Perrot; Duchesnay, E.. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011

[18] Bishop, Christopher. *Pattern Recognition and Machine Learning*. Springer.

[19] Torres, Delfim. (2020). *Programação Matemática*.

[20] Awad, Mariette; Khanna, Rahul. (2015). Support Vector Regression.

[21] Sapankevych, N.; Sankar, R.. (2009). Time Series Prediction Using Support Vector Machines: A Survey. *IEEE Computational Intelligence Magazine*, 4.

[22] Rivas, Pablo. (2013). Support Vector Machines for Regression: A Succinct Review of Large-Scale and Linear Programming Formulations. *International Journal of Intelligence Science* 03. 5-14.

[23] Shewhart, A.Walter; S.Wilks, Samuel. (2010). *Time Series Application to Finance with R and S-Plus*. 2º edition. Wiley.

[24] Murteira, B.; Muller, D.A.; Turkman, K.F.. (1993). *Análise de Sucessões Cronológicas*. McGraw-Hill, Lisboa.

[25] Hyndman, R.; Koehler, A.B.; Ord, J.K.; Snyder, R.D. (2008). *Forecasting with Exponential Smoothing: The State Space Approach*. Springer.

[26] Minsky, Marvin; Papert, Seymour. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press.

[27] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6, 386-408 .

[28] Widrow, B. (1960). An adapative ADALINE neuron using chemical memistors.

[29] Rumelhart, D.; Hinton, G.E.; Williams, R.J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533-536.

[30] Duchi, J.C., Hazan, E., Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res..*

[31] Kingma, D.P.; Ba, J. (2015). Adam: A Method for Stochastic Optimization. *CoRR, abs/1412.6980.*

[32] Ramachandran, P.; Zoph, B.; Le, Q.V. (2018). Searching for Activation Functions. *ArXiv, abs/1710.05941.*

[33] Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron. (2016) .*Deep Learning*. MIT Press

[34] Hochreiter, S.; Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9, 1735-1780.

[35] Henkelmann, Robby. (2018). *A Deep Learning based Approach for Automotive Spare Part Demand Forecasting*. Otto-von-Guericke-Universität Magdeburg.

[36] Ribeiro, J.A. (2017). Forecasting the demand of spare components for a better stock management: a case study at Portugalia Airlines.

[37] Klug, Florian. (2010). *Logistikmanagement in der Automobilindustrie.* 10.1007/978-3-642-05293-4.

[38] Costa, Raquel. (2015). *Previsão de Vendas Aplicada a Perfis de Alumínio.*

[39] Costa, M.; Gonçalves, A.; Silva, J. (2015). Forecasting time series combining Holt-Winters and bootstrap approaches.

[40] Wang, J.; Pan, X.; Wang, L.; Wei, W. (2018). Method of Spare Parts Prediction Models Evaluation Based on Grey Comprehensive Correlation Degree and Association Rules Mining: A Case Study in Aviation. *Mathematical Problems in Engineering*, 2018, 1-10.

[41] Vargas, C.A.; Cortes, Mayra. (2017). Automobile spare-parts forecasting: A comparative study of time series methods. *International Journal of Automotive and Mechanical Engineering.* 14. 3898-3912.

[42] Kontrec, Nataša; Panić, Stefan. (2017). Spare Parts Forecasting Based on Reliability | InTechOpen.

[43] Dickey, D.A.; Fuller, W.A. (1981). Likelihood ratio statistics for autoregressive time series with a unit root. *Econometrica* 49 1057-1072

[44] Evans, G. B. A.; Savin, N. E. (1984). Testing for unit roots: 2. *Econometrica* 52 1241-1270.

[45] Wolters, Juergen; Hassler, Uwe. (2006). Unit root testing. *AStA Advances in Statistical Analysis*. 90. 43-58. 10.1007/s10182-006-0220-6.

[46] Vladimir, Vapnik. (1995). *The Nature of Statistical Learning Theory.* Springer.

[47] Hu, Wenqing. (2020). Nonlinear Optimization in Machine Learning:A Series of Lecture Notes at Missouri ST. Missouri University of Science and Technology.

[48] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *ArXiv, abs/1609.04747.*

[49] LeCun, Y.; Haffner, P.; Bottou, L.; Bengio, Y. (1999). Object Recognition with Gradient-Based Learning. *Shape, Contour and Grouping in Computer Vision.*

[50] Krizhevsky, A.; Sutskever, I.; Hinton, G.E. (2012). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60, 84 - 90.

[51] Fukushima, K. .(1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4): 93-202.

[52] Taylor, Sean J.; Letham, Benjamin. (2018) Forecasting at Scale. *The American Statistician*, 72:1, 37-45, DOI: 10.1080/00031305.2017.1380080

[53] Jensen, C.S.; Pedersen, T.; Thomsen, C. (2010). *Multidimensional Databases and Data Warehousing.*

[54] Kannan, Bhuvana; Kodi, Ganesh; Padilla, Oscar; Gray, Dough; Smith, Barry C. (2020). Forecasting Spare Parts Sporadic Demand Using Traditional Methods and Machine Learning - a Comparative Study. *SMU Data Science Review*

Apêndices

Apêndice A

sbroETL-library

```yaml
library:
  exec:
    convert_vendas:
      conn_dest: conn01
      function_file: true
      params:
        N_filter: 2
        SeasonalDecompose: false
        alpha: 2
        filename: ../01-data/vendas.csv
        freq: 52
        k: 5
    data_raw:
      conn_dest: conn53
      function_file: true
      params:
        filename: ../01-data/vendas.csv
    ts01_Autocorrelation:
      conn_sources: conn01
      function_file: true
    ts03_HoltWinters_train:
      conn_dest: conn02
      conn_sources: conn01
      function_file: true
      params:
```

```
      seasonal: add
      seasonal_periods: 52
  ts07_ARIMA_train:
    conn_dest: conn02
    conn_sources: conn01
    function_file: true
    params:
      alpha: 0.05
      error_action: ignore
      m: 52
      seasonal: true
      supress_warnings: true
  ts11_Prophet_train:
    conn_dest: conn02
    conn_sources: conn01
    container_func: true
    container_name: prophet
    debug: false
    params:
      alpha: 0.05
      daily_seasonality: false
      test_freq: W
      weekly_seasonality: false
      yearly_seasonality: true
  ts13_Default_train:
    conn_dest: conn02
    conn_sources: conn01
    function_file: true
    params:
      model: prev_value
      prev_shift: 52
  ts15_MLP_train:
    conn_dest: conn02
    conn_sources: conn01
    container_func: true
    container_name: tensorflow
    debug: false
```

```
    params:
      batch_size: 52
      epochs: 1000
      layer1_func: swish
      layer1_size: 156
      layer2_func: relu
      layer2_size: 156
      loss_function: mse
      nprev: 3
      optimizer: adam
      steps: 3
  ts17_SVR_train:
    conn_dest: conn02
    conn_sources: conn01
    function_file: true
    params:
      nprev: 3
      steps: 3
      kernel: ['linear', 'rbf','poly']
      C: [0.01,1, 100]
      epsilon: [0.1,0.2,0.5,0.3]
  ts19_LSTM_train:
    conn_dest: conn02
    conn_sources: conn01
    container_func: true
    container_name: tensorflow
    debug: false
    params:
      batch_size: 52
      epochs: 1000
      layer1_func: relu
      layer1_size: 156
      loss_function: mse
      n_features: 1
      nprev: 3
      optimizer: adam
      steps: 3
```

```
    ts21_CNN_train:
      conn_dest: conn02
      conn_sources: conn01
      container_func: true
      container_name: tensorflow
      debug: false
      params:
        activation: relu
        batch_size: 52
        epochs: 1000
        filters: 156
        kernel_size: 2
        loss_function: mse
        n_features: 1
        nprev: 3
        optimizer: adam
        pool_size: 2
        steps: 3
    ts23_XGBoost_train:
      conn_dest: conn02
      conn_sources: conn01
      function_file: true
      params:
        nprev: 3
        steps: 3
        booster: ['gbtree']
        objective: ['reg:squarederror']
        learning_rate: [0.3]
        max_depth: [6]
        min_child_weight: [1]
        silent: [1]
        subsample: [1]
        colsample_bytree: [1]
        n_estimators: [100]
        alpha: [0]
        tree_method: ['hist']
    ts99_modelStacking_train:
```

```
      conn_dest: conn02
      conn_sources: conn02
      container_func: true
      container_name: tensorflow
      debug: false
      params:
        activation: relu
        batch_size: 3
        epochs: 1000
        loss: mse
        neurons: 32
        optimizer: adam
        n_splits: 3
        n_repeats: 10
        random_state: 1
  global_params:
    best_metric: rmse6040
    cross-validation: --,0,:3
    graphs: false
    pred_lower_cut: 0.0
```