Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Modelação Estatística e Análise de Dados

Área de especialização | Modelação Estatística e Análise de Dados

Dissertação

Estudo comparativo de métodos de análise e previsão de séries temporais: métodos estatísticos versus redes neuronais

Fernando Jorge Tavares dos Santos Moreno

Orientador(es) | Dulce Maria de Oliveira Gomes

Évora 2021

Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Modelação Estatística e Análise de Dados

Área de especialização | Modelação Estatística e Análise de Dados

Dissertação

Estudo comparativo de métodos de análise e previsão de séries temporais: métodos estatísticos versus redes neuronais

Fernando Jorge Tavares dos Santos Moreno

Orientador(es) | Dulce Maria de Oliveira Gomes

Évora 2021

A dissertação foi objeto de apreciação e discussão pública pelo seguinte júri nomeado pelo Diretor da Escola de Ciências e Tecnologia:

Presidente	|	Manuel Joaquim Piteira Minhoto (Universidade de Évora)
Vogais	|	Carlos Correia Ramos (Universidade de Évora) Dulce Maria de Oliveira Gomes (Universidade de Évora)

Évora 2021

"Não fiquem assim, preocupados. Vocês sempre creram em Deus, creiam agora em mim. Na casa de meu Pai há muitas moradas; se não fosse assim, eu vo-lo teria dito; vou preparar-vos lugar. E depois que eu for e terminar a preparação desse lugar, virei outra vez, e vos levarei para mim mesmo para que onde eu estiver estejais vos também. E já sabeis para onde vou, e sabeis o caminho.

João. 14:1 a 4

Agradecimentos

Agradeço a Deus, por mais uma etapa na minha vida, ultrapassando todos os desafios que foram surgindo.

Agradeço a minha mãe (Maria Fernanda Mascarenhas) pelo cuidado e preocupação, longe mas sempre perto, ao meu irmão José Maria Moreno, pela sempre atenção, ajuda e alegria compartilhada nos meus sucessos académicos, ao meu irmão Osvaldo Moreno, pela força e coragem que possui que é uma inspiração para mim, e a todos os meus irmãos e familia, pelo suporte concedido.

Agradeço a minha orientadora (Dulce Maria Gomes) pelos apoios constantes e pela sempre disponibilidade durante todo o trabalho para me ajudar a ultrapassar as dificuldades, mas também pela orientação que me possibilitaram concluir este trabalho.

Agradeço a todos os meus professores de mestrado, que me ajudaram durante os dois anos e sempre com muita disponibildade e profissionalismo. Dois anos que me abriram novos horizontes no meu futuro académico e profissional.

Aos meus colegas, Ana Ferrari, Finorio Castigo e Rodrigo Silva, um muito obrigado, pelos diversos trabalhos que fizemos juntos, com ajudas mutuas.

As minhas amigas, Helena Mendes, Alice Cebola, Anita que sempre me trataram com muito carinho e simpatia, e ao meu amigo Eduardo que por muitas vezes estivemos juntos. E a todas e todos que de uma forma ou outra me possibilitaram a conclusão desse mestrado.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AR Auto-regressivo

ARIMA Auto-Regressivo Integrado e Médias Móveis

SARIMA Auto-Regressivo Integrado e Médias Móveis com Sazonalidade

FAC Função de auto-correlação

FACP Função de auto-correlação parcial

MAPE Erro percentual médio absoluto

ME Erro médio

RMSE Raiz quadrada média do erro

MAE Erro/Desvio médio absoluto

MPE Erro médio percentual

RNN Redes neuronais recorrentes

LSTM Rede de memória de longo prazo

GRU Unidades recorrentes com portão

Sumário

Os mecanismos modernos de recolha de dados, não só tem levado ao surgimento de novas técnicas de análise e compreensão do comportamento dos dados, mas também a dúvidas sobre as melhores técnicas a utilizar. Neste contexto, as séries temporais assumem particular importância, visto que a compreensão da sequência dos acontecimentos possibilita a determinação de critérios de análise e previsão de qualquer problema sequencialmente disposto no tempo.

A compreensão dos acontecimentos fornece para os diferentes intervenientes uma vantagem num ambiente de maior competitividade, e esse trabalho vai na prespetiva de esclarecer as diferentes técnicas de análise e ajustamentos dos dados sequênciais, tanto utilizando métodos estatísticos como técnicas de inteligência artificial.

Primeiramente, efectuou-se uma revisão bibliográfica tanto das técnicas estatísticas como de inteligência artificial posteriormente houve uma divisão dos dados entre séries não estacionárias com tendência e séries não estacionárias com tendência e sazonalidade, e aplicadas diferentes técnicas para o ajustamento das séries, e feita no final uma comparação entre elas. O resultado deste trabalho, vai no sentido de fornecer os mecanismos existentes de ajustamento e servir como referência na utilização de análise de dados sequênciais, e abrir oportunidades de novas prespetivas de análise de dados.

Palavras chave: Séries temporais, Redes Neuronais, tendência, estacionariedade, sazonalidade

Abstract

Modern data collection mechanisms have not only led to the emergence of new techniques for analyzing and understanding data behavior, but also to doubts about the best techniques to use. In this context, the time series are of particular importance, since understanding the sequence of events, makes it possible to determine criteria for analyzing and predicting any problem sequentially arranged over time.

Understanding the events provides different players with an advantage in a more competitive environment, and this work aims to clarify the different techniques for analyzing and adjusting sequential data, using both statistical and artificial intelligence methods.

First, a bibliographic review of both statistical and artificial intelligence techniques was carried out, later on there was a division of the data between non-stationary series with tendency and non-stationary series with trend and seasonality, and different techniques were applied to adjust the series, and a comparison is made at the end. The result of this work is to provide the existing adjustment mechanisms and to serve as a reference in the use of sequential data analysis, and to open opportunities for new data analysis perspectives.

Keywords: Time series, Neural Networks, trend, stationary, seasonality

1

Introdução

O conhecimento dos dados sequenciais se revela de grande importância para a compreensão da evolução dos diferentes fenómenos no tempo. A análise destes requer o conhecimento dos mecanismos corretos e mais apropriados para que estas análises possam ser o melhor possível, pressupondo um adequado ajustamento entre o tipo dos dados e a técnica de análise a utilizar.

Existem diversas técnicas utilizando diversos modelos estatísticos de análise de séries temporais, tais como, os modelos clássicos de decomposição como o metodo X11, Tramo/Seats, tendência e sazonalidade utilizando LOESS, Holt e Holt-Winter, modelos lineares de análise de séries temporais como o modelo auto-regressivo (AR), auto-regressivo integrado de médias moveis (ARIMA) ou o modelo auto-regressivo integrado de médias móveis com sazonalidade (SARIMA), mas também técnicas mais recentes de análises utilizando a inteligência artificial, com a utilização apropriada de algoritmos de aprendizagem de máquina, nomeadamente as redes neuronais recorrentes, como o algoritmo das redes neuronais recorrentes (RNN), o algoritmo das redes neuronais com unidades recorrentes com portão (GRU) ou as redes de memória de longo prazo (LSTM).

Esses modelos foram apresentados e utilizados para a análise dos diferentes tipos de dados, com tendência crescente, com tendência decrescente, com tendência e sazonalidade crescente e com tendência e sazonalidade decrescente.

Quanto as diferentes bases de dados utilizadas, algumas apresentam muitas oscilações com tendência crescente como a produção de carros em periodos trimestrais no Reino Unido e temperatura de corpo de um castor feminino medidas a cada 10 minutos, outros como medições anuais do nível, em pés, do Lago Huron, e o preço de uma dúzia de ovos nos Estados Unidos apresentam tendências decrescentes, enquanto outros ainda apresentam ter tendência e sazonalidade como a venda de passe em lisboa e as passagens aéreas no Reino Unido. Outras séries não apresentam muitas oscilações, mas onde apresentaram ser dados com tendência, como a série morte nos Estados Unidos, Consumo Público em Portugal, o Produto Interno Bruto português.

1.1 Motivação e relevância do trabalho

Existem diferentes técnicas e métodos possíveis, de ajustamento e previsão para dados sequenciais, isso tanto através dos metodos tradicionais como através dos mecanismos de inteligencia artificial. Muitas duvidas se levantam quanto a aplicação de uma determinada técnica, em detrimento da outra, isso tanto entre os metodos tradicionais e as técnicas de redes neuronais como mesmo entre as técnicas de redes neuronais em si. Quando e porque utilizar uma ou outra? Qual é o desempenho de cada técnica, para os diferentes tipos de séries? Quais são os fatores que determinam essa utilização? Estas são algumas das questões que levaram a efetivação deste trabalho, na busca de uma resposta clara e coerente.

Com este estudo pretende-se, desse modo contribuir para uma análise dos mecanismos necessários ao ajustamento e previsão utilizando as abordagens tradicionais por exemplos os modelos ARIMA e de decomposição classica, e as abordagens com as redes neuronais, nomeadamente o perceptron de multiplas camadas (MLP), as redes neuronais recorrentes (RNN), as redes recorrentes com memória de longo prazo (LSTM) e a unidade recorrente de porta (GRU).

1.2 Estruturação do trabalho

O trabalho está organizado em 10 capítulos dispostos como segue.

No Capítulo 1, faz-se uma introdução ao tema em estudo e descreve-se a motivação do mesmo.

No Capítulo 2, é abordada uma primeira definição das séries temporais, fazendo a diferença entre as séries regulares e não regulares, introduziu-se as noções de autocorrelação, autocorrelação parcial, series estacionarias e não estacionarias e processo estocasticos.

No Capítulo 3, apresentam-se os diferentes modelos de decomposição de séries temporais, e as respetivas vantagens e desvantagens.

O Capítulo 4, apresentam-se os modelos lineares de séries temporais, diferenciando-os dos clássicos, e aborda-se as etapas de modelação utilizando a metodologia Box-Jenkins.

Dos capítulos 5-8 irá debruçar-se sobre as várias abordagens de redes neuronais e sua história, e por ser um tema relativamente novo e de aplicação residual, em particular na área das Séries Temporais, serão dedicado estes capítulos.

O Capítulo 6, apresenta o conceito de aprendizado de máquina e alguns conceitos inerentes.

No Capítulo 7, consta os fundamentos da aprendizagem profunda, como se podem otimizar e treinar uma rede neuronal.

O Capítulo 8, apresenta as diferentes arquiteturas das redes neuronais abordadas, *perceptron* de múltiplas camadas (MLP), as redes neuronais recorrentes (RNN), as redes recorrentes com memória de longo prazo (LSTM) e a unidade recorrente de porta (GRU).

O Capítulo 9, apresenta os principais resultados do estudo para o ajustamento e para a previsão.

O Capítulo 10, compreende as considerações finais, as limitações do trabalho e as recomendações para estudos futuros.

2

Séries temporais

Pode-se definir de uma forma informal uma série temporal como um conjunto de observações x_t, cada uma delas observada num instante particular t. Nesse caso nas séries temporais o momento em que é coletada cada observação necessita ser registado e é utilizado na análise, modelação e previsão da série e, de onde vem o conceito de séries temporais.

Observações podem ser coletados a intervalos regulares, estas correspondendo as séries cuja as observações estão igualmente espaçados ao longo do tempo, ou seja, existe um mesmo espaço temporal entre duas observações consecutivas quaisquer, diariamente, semanalmente, mensalmente, trimestralmente, anualmente, a cada cinco anos, ou decenalmente (Torgo, 2009). Com o surgimento dos computadores de alta velocidade, os dados agora podem ser coletados a intervalos extremamente curtos, como os relativo a bolsas de valores, ou a intervalos irregulares onde não tem necessariamente o mesmo espaçamento temporal (Gujarati, 2011).

2.1 Processos estocásticos

Segundo Morettin e Toloi (2006), os métodos aplicados para descrever séries temporais são considerados processos estocásticos, ou seja, processos cuja evolução no tempo é gerada e controlada por leis probabilísticas.

Seja T um conjunto aleatório. Um processo estocástico é uma família $Z = Z(t), t \in T$, tal que, para cada $t \in T, Z(t)$ é uma variável aleatória.

Nesse sentido, um processo estocástico será uma família de variáveis aleatórias, que se supoem ser definidas num mesmo espaço de probalidades. O conjunto T é normalmente tomado como o conjunto dos inteiros $\mathbb{Z}$=$\{0, 1, 2, 3, ...n\}$ ou o conjunto dos reais $\mathbb{R}$. Também, para cada $t \in T$, $Z(t)$ será uma variável aleatória real. Como, para $t \in T$, $Z(t)$ é uma variável aleatória definida sobre um espaço amostral universal Ω, que é o conjunto de todos os resultados possíveis, na realidade $Z(t)$ é uma função de dois argumentos, $Z(t, w), t \in T, w \in \Omega$.

Por outro lado, para cada $w \in \Omega$ fixado, obteremos uma função de t, ou seja, uma realização ou trajetória do processo, ou ainda, uma série temporal.

2.1.1 Processos estocásticos estacionários

Um processo estocástico será chamado de estacionário se sua média e variância forem constantes ao longo do tempo e o valor da covariância entre os dois períodos de tempo depender apenas da distância, do intervalo ou da desfasagem entre os dois períodos. Esse processo estocástico é conhecido como processo estocástico fracamente estacionário, ou estacionário em covariância ou ainda de segunda ordem (Gujarati, 2011).

Considera-se Y_t como uma série temporal estocástica com as seguintes propriedades:

$$\text{Média} : E(Y_t) = \mu \tag{2.1}$$

$$\text{Variância} : Var(Y_t) = \sigma^2 \tag{2.2}$$

$$\text{Covariância} : \gamma_k = E[(Y_t - \mu)(Y_{t+k} - \mu)] \tag{2.3}$$

em que γ_k , a covariância na desfasagem k, é a covariância entre os valores de Y_t e Y_{t+k}, isto é, entre dois valores de Y separados por k. Se $k = 0$, obtemos γ_0, que é simplesmente a variância de Y_t se $k = 1$, γ_1 é a covariância entre os dois valores adjacentes de Y.

Suponha que mudemos a origem de Y de Y_t para Y_{t+m} . Agora, se Y_t for estacionário, a média e a variância de Y_{t+m} deverão ser iguais àquelas de Y_t.

Em resumo, se uma série temporal for estacionária, não importa em que ponto medirmos a média e a variância, vão sempre permanecer as mesmas; isto é, elas serão invariantes no tempo, e a covariância depende do intervalo de tempo. Tal série temporal tenderá a retornar para a sua média (o que chamamos

de reversão da média), e flutuações em torno dessa média (medida por sua variância) terão, de modo geral, uma amplitude constante e a autocovariância (que é a covariância de uma variável com sua desfasagem) depende do intervalo entre os tempos (Gujarati, 2011). Em outras palavras, um processo estacionário não se desviará muito de seu valor médio em virtude da variância finita.

Um processo fundamental na construção dos modelos de séries temporais é o ruído branco. Um processo estocástico e_t é um ruído branco se tiver média constante, variância constante e não for correlacionado a qualquer realização da própria série (autocorrelação igual a zero). Normalmente considera-se a média nula dado ser o conveniência, pois seria nesse caso possível especificar um ruído branco cuja média fosse diferente de zero. Entretanto, pode-se centrar em zero tal série, sem prejuízo de suas demais propriedades (Bueno, 2012).

Uma série temporal pode também ser fortemente (ou estritamente) estacionária se as funções de distribuição conjuntas são idênticas.

2.1.2 Processo estocásticos não estacionários

Uma grande maioria das séries temporais apresentam algum tipo de não estacionariedade. Podemos apresentar a tendência (crescente ou decrescente) e série não estacionárias homogeneas, quando a série é estacionária, por um periodo de tempo, em um certo nível, mudando para outro nível por outro período de tempo, assim sucessivamente.

Quando uma série temporal não é estacionária, não se pode estimá-la normalmente, não sendo possível nesse caso estimar todos os momentos da série. Em particular, é impossível efetuar inferências estatísticas (Bueno, 2012). A variância não condicional de um processo auto-regressivo (AR) de ordem 1, que será introduzido mais adiante é:

$$Var(Y_t) = \frac{1}{1-\phi^2} \tag{2.4}$$

Para $\phi = 1$, o que caracteriza uma série não estacionária de raiz unitária, a variância explode. Todavia, pode-se resolver esse problema. Em séries univariadas, basicamente se diferencia a série tantas vezes quantas sejam necessárias para estacionarizá-la. O caso mais comum é encontrar uma série com raiz unitária, de forma que basta a primeira diferença dessa série para estacionarizá-la, entretanto mais diferenças poderão ser necessárias.

- **Transformação de *Box-Cox***

  Uma transformação *Box-Cox* tem por objetivo estabilizar a variância. No centro da transformação de *Box-Cox* está um expoente, lambda (λ), que usualmente se considera a variar entre -5 e 5. Todos os valores de $\lambda \in$ R são considerados e o valor ideal para seus dados é selecionado; O "valor ótimo" é aquele que minimiza a variância. A transformação de Y tem a forma:

$$\begin{cases} T_\lambda(Y_t) = \frac{Y^\lambda - 1}{\lambda}, & se\ \lambda \neq 0; \\ log(Y_t), & se\ \lambda = 0. \end{cases} \tag{2.5}$$

- **Diferenciação** A diferenciação consiste em remover os sinais de tendências, estabilizando a média, enquanto que a transformação de Box-Cox tem por objetivo reduzir a variância. A diferenciação

simples a um passo ($d = 1$) é a diferença do valor do período t com o valor do período anterior $t-1$. E a diferenciação a dois passos (d=2), representa-se usualmente por:

$$D^2Y_t = D(DY_t)$$

$$D^2Y_t = D(Y_t - Y_{t-1})$$

$$D^3Y_t = D^2(Y_{t-1} - Y_{t-2})$$

ou seja, é a diferenciação simples da diferenciação simples, tudo a 1 passo ($d = 1$).

A presença de tendências determinísticas ou estocásticas geram processos bastante semelhantes. Todavia a maneira de tratar a não estacionariedade em ambos os casos é diferente, e aplicar o método incorreto pode causar sérios danos à informação contida na série temporal, pois muda muito o processo resultante (Ferreira, 2018).

A diferenciação remove tanto tendências estocásticas quanto determinísticas. Se diferenciarmos um processo tendência estacionário $Y_t = \alpha + \beta t + \epsilon_t$, obtemos $DY_t = \beta + \epsilon_t - \epsilon_{t-1}$, que é um processo de média móvel de ordem 1, MA(1), somado a uma constante, que é estacionário (Cowpertwait e Metcalfe, 2009).

Pode-se regredir os dados numa sequência representando o tempo e tomarmos o resíduo como nossa nova série temporal. Tal processo é igualmente estacionário, porém preserva mais da estrutura original removendo apenas uma componente deterministica. Todavia, para uma série que apresenta não estacionariedade devido a presença de uma raiz unitária, a remoção de tendência não a torna estacionária (Ferreira, 2018).

Por exemplo, se tomarmos um passeio aleatório puro e removermos a tendência da série, a série continua não estacionária dado que continua com o passeio aleatório. Mas, vemos que a remoção de tendência de um processo tendência-estacionário é exatamente o que buscamos em termos da função de autocorrelação parcial (FAC), que será abordada mais adiante. Exatamente por tais ambiguidades, faz-se necessário uma metodologia formal para a análise de estacionariedade, sendo correntemente os testes hipótese baseados na existência de raizes unitárias os mais utilizados.

2.2 Autocorrelação e autocorrelação parcial

Se $\{Y_t; t \in \mathbb{Z}\}$ for estacionária, então $E[Y_t]$ é constante e $Cov(Y_t, Y_s)$ será função apenas de $[t-s]$, para todo o $s, t \in$ Z; torna-se assim especialmente importante conhecer:

$$\mu = E[Y_t] \tag{2.6}$$

$$\gamma_k = Cov(Y_t, Y_{t+k}), k \in \mathbb{Z} : \text{função de autocovariância de } \{Y_t; t \in \mathbb{Z}\} \tag{2.7}$$

Em virtude de a função de autocovariância ser sensível às unidades em que são medidas as observações, utiliza-se também a função de autocorrelação, $\{\gamma_k; k \in \mathbb{Z}\}$ dada por:

$$\rho_k = \frac{\gamma_k}{\gamma_0} = \frac{Cov(Y_t, Y_{t+k})}{Var(Y_t)},\ k \in \mathbb{Z}. \tag{2.8}$$

A função de autocorrelação (ρ_k) permitirá identificar a ordem q de um processo MA. As funções de autocorrelação de alguns processos já conhecidos são (Bueno, 2012):

Modelo	FAC
MA(q)	$\rho = \frac{\theta_j + \theta_j + 1\theta_1 + \theta_j + 2\theta_2 + ... + \theta_q\theta_q - j}{\sum_{j=0}^{q} \theta_j^2}$ $j = 1, 2, ..., q$
AR(1)	$\rho_j = \phi^j$, j= 1,2,....
AR(p)	$\rho_j = \phi_1\rho_{j-1} + \phi_2\rho_{j-2} + ... + \phi_p\rho_{j-p}$, j= 1,2,..
ARIMA(1,1)	$\begin{cases} \rho_1 = \frac{(1+\phi_1\theta_1)(\phi_1+\theta_1)}{1+\theta_1^2+2\phi_1\theta_1} \\ \rho_j = \phi_1\rho_{j-1}, j > 1 \end{cases}$

A tabela anterior mostra que a autocorrelação do MA torna-se zero a partir da desfasagem q, enquanto a dos outros modelos decai, sendo que esse decaimento é exponencial no caso do AR(1), e pode ter várias configurações em um ARMA (p, q).

A função de autocorrelação parcial e a autocorrelação entre Y_t e Y_{t+k}, conhecidos todos os pontos entre t e $t + k$, definida por:

$$\phi(k,k) = \frac{cov(Y_t, Y_{t+k}|Y_{t+1}, ..., Y_{t+k-1})}{\sqrt{var[Y_t|Y_{t+1}, ..., Y_{t+k-1}] \cdot var[Y_{t+k}|Y_{t+1}, ..., Y_{t+k-1}]}} \tag{2.9}$$

Existem diferentes formas de modelação de uma mesma série temporal, dado a dificuldade em inferir qual é o padrão gerador daquela função. Na prática, o que é importante é termos um resíduo que seja estatisticamente um ruído branco, para tal será necessário máximo de informações da série. Essa seria a melhor modelação possível de ser obtida a partir dos dados observados. Supostamente, essa modelação levaria às melhores previsões estatísticas (Bueno, 2012).

Em um modelo AR(1), existe uma correlação implícita entre Y_t e Y_{t-2}. Isso está presente na FAC, por meio do decaimento exponencial. No entanto, é possível escolher as correlações, de forma a manter-se apenas a correlação pura entre duas observações. Essa filtragem leva a uma função de autocorrelação parcial (FACP), pela qual eliminam-se as correlações implícitas entre duas observações. Dessa forma, em um AR(1), a autocorrelação parcial entre Y_t e Y_{t-2} desaparece (Bueno, 2012).

A função de autocorrelação parcial é estimada a partir das seguintes regressões em que a série original tem sua média subtraída:

$$Y_t = \phi_{j,1}Y_{t-1} + \phi_{j,2}Y_{t-2} + ... + \phi_{j,j}Y_{t-j} + e_t \tag{2.10}$$

em que e_t é um erro.

O procedimento consiste então, em regredir Y_t contra Y_{t-1} e obter $\hat{\phi}_{1,1}$. Em seguida, deve-se regredir y_t contra Y_{t-1} e Y_{t-2} · São obtidos dessa forma os coeficientes $\hat{\phi}_{2,1}$ e $\hat{\phi}_{2,2}$ dos quais interessa apenas este último, e assim por diante.

Se a análise recair, por exemplo, sobre um modelo AR(2), os coeficientes obtidos a partir de $j > 2$ deverão

ser iguais a zero. Genericamente, em um AR(p), serão encontrados coeficientes diferentes de zero até $\hat{\phi}_{1,1}$ e estatisticamente iguais a zero a partir de então.

Em um MA(q), dada a condição de invertibilidade que torna esse processo um AR(∞), pode-se mostrar que os coeficientes $\hat{\phi}_{j,j}$ decaem. Quando se tem um modelo ARMA(p, q), há decaimento a partir da desfasagem p cuja configuração depende da magnitude dos parâmetros.

2.3 Séries temporais estacionárias e não estacionárias

Perante uma série temporal a primeira visão através de um gráfico é de grande importância para se ter uma ideia da evolução dos dados. Através de uma análise gráfica pode-se ter a ideia da estacionariedade ou não da série, mas é insuficiente para verificar se será estacionária, para tal deve-se complementar com testes estatísticos apropriados. Alguns dos testes mais utilizados são os testes de Dickley-Fuller Aumentado (ADF), teste de Phillips-Perron e teste de Kwiatkowski–Phillips–Schmidt–Shin (KPSS), (Morettin e Toloi, 2006).

As séries temporais apresentam algumas características tais como:

Tendência:
Existe uma tendência quando há um aumento ou diminuição a longo prazo nos dados, não precisando ser linear. Por vezes, nos referimos a uma tendência como "mudança de direção", quando ela pode passar de uma tendência crescente para uma tendência decrescente, (Hyndman, 2018).

A ideia essencial da tendência é que ela represente características suaves da série temporal. Na prática, isso significa que gostaríamos de representá-lo por uma função contínua do tempo. Existem várias classes de funções que podem ser apropriadas em circunstâncias específicas, como polinómios, ondas sinusoidais ou funções harmónicas e funções lineares por partes (Kendall & Ord, 1993).

Ciclo:

Um ciclo ocorre quando a exibição de dados aumenta e diminui sem frequência fixa. A duração dessas flutuações depende da unidade temporal mas é geralmente de pelo menos 2 anos, (Hyndman, 2018).

Sazonalidade:

Um padrão sazonal ocorre quando uma série temporal é afetada por fatores sazonais, como a época do ano ou o dia da semana. A sazonalidade é sempre de uma frequência fixa e conhecida (Hyndman, 2018).

Segundo (Kendall & Ord, 1993), os efeitos sazonais, embora possam variar um pouco no tempo médio de ocorrência durante o ano, têm um grau de regularidade que outros elementos da série temporal geralmente não apresentam. No domínio do tempo, é impossível determinar os efeitos sazonais sem alguns ajustes anteriores para a tendência. Assim existem várias razões para observar efeitos residuais após a remoção da tendência:

1. Comparar uma variável em diferentes pontos do ano como um fenómeno puramente intra-anual;
2. Remover efeitos sazonais das séries para estudar seus outros constituintes não "contaminados"pelo componente sazonal;
3. Para "Corrigir"um valor atual para efeitos sazonais

3

Modelo clássico de decomposição das séries temporais

A compreensão de uma série temporal passa pela análise detalhada de cada componente que dela faz parte. Em 1930, Macauley, concetualizou o conceito da decomposição clássica introduzindo o métodos das médias móveis para ajustes sazonais de uma série, consistindo em cálculo das componentes sazonais, estimação da tendência e a divisão da média móvel pela estimativa da tendência. A partir do método de decomposição clássica outros métodos foram surgindo, todos estes baseados na decomposição clássica de séries temporais (aditiva ou multiplicativa). O modelo aditivo e descrito como:

$$Y = T + S + I$$

e o modelo multiplicativo e representado por

$$Y = T * S * I$$

onde Y é a série analisada, T é a componente tendência, S é a componente sazonal e I é a componente irregular. Esses métodos de decomposição sao o método X11, decomposição pelo método da Regressão

de série temporal com ruído ARIMA, valores ausentes e outliers (TRAMO) e Extração de sinal em série temporal ARIMA (SEATS), normalmente usam-se juntos como sendo TRAMO/SEATS, decomposição sazonal e tendência usando LOESS (STL), modelo de suavização exponencial, cada uma fundamentada em princípios que os distinguem, levando a diferenciar entre as vantagens e desvantagens de cada um dentre eles.

3.1 Decomposição pelo método X11

O método X11 teve a sua origem no método clássico de decomposição de séries temporais, baseando-se em médias móveis. Esse método faz parte dos métodos não paramétricos, que permitem decompor a série em componentes não observáveis mediante procedimentos iterativos baseados em filtragens sucessivas, onde se remove a sazonalidade para estimar a série ajustada sazonalmente.

A decomposição da série temporal pelo método X11 pode ser aditiva ou multiplicativa; sendo que a multiplicativa se torna aditiva a partir da transformação logarítmica. O programa usa dois tipos diferentes de médias móveis com o propósito de estimar a tendência e a componente sazonal. Primeiro, a componente de tendência é removida e, em seguida, a sazonalidade. Feita a decomposição da série temporal a partir do método X11, a série dessazonalizada é estimada. No caso de um modelo aditivo, a série sem sazonalidade é obtida subtraindo-se a componente sazonal da série original. Já no caso de um modelo multiplicativo, a série com ajuste sazonal é obtida dividindo-se a série original pela componente sazonal estimada, (Shiskin et al. 1967).

Fundamentada no método de decomposição clássica, tem, no entanto diversas características que o diferenciam deste ultimo para superar as desvantagens relativas, nomeadamente as estimativas do ciclo de tendência estão disponíveis para todas as observações, e o componente sazonal pode variar lentamente ao longo do tempo. O X11 também possui alguns métodos sofisticados para lidar com a variação dos dias ou efeitos de férias. O processo é totalmente automático e tende a ser altamente robusto para discrepâncias e mudanças de nível na série temporal, (Hyndman, 2018).

No entanto, não é apropriado para todos os tipos de sazonalidade, sendo mais apropriado para dados mensais e trimestrais.

3.2 Decomposição pelo método TRAMO/SEATS

Esses programas foram desenvolvidos por Victor Gómez e Agustín Maravall no Banco da Espanha. Ambos os programas foram desenvolvidos para trabalharem juntos. Utilizando um modelo de regressão com processo ARIMA o TRAMO faz um pré-ajuste da série removendo diversos efeitos determinísticos. O SEATS executa o ajuste sazonal. O TRAMO funciona com dados trimestrais e mensais.

As principais aplicações são a previsão, ajuste sazonal, estimativa de ciclo de tendência, detecção e correção de outliers, estimativa de efeitos especiais e controle de qualidade de dados.

O programa decompõe a série temporal em suas componentes não observáveis utilizando filtros derivados do modelo ARIMA passados pelo TRAMO. Caso *outliers* e efeitos determinísticos não tenham sido inclusos no modelo e não existam observações faltantes, o SEATS pode ser executado sozinho, uma vez que há implementado nele a mesma rotina do TRAMO para a obtenção do modelo ARIMA (European Commission Grant, 2007).

3.3 STL - *seasonal and trend using* LOESS

As vantagens do STL é que ele pode identificar componentes sazonais que mudam com o tempo, responde a tendências não-lineares e é robusto na presença de *outliers*.

O STL tem várias vantagens sobre os métodos de decomposição clássicos, SEATS e X11:

- Diferentemente do SEATS e do X11, o STL manipula qualquer tipo de sazonalidade, não apenas os dados mensais e trimestrais.
- O componente sazonal pode mudar ao longo do tempo e a taxa de alteração pode ser controlada pelo utilizador.
- A suavidade do ciclo de tendências também pode ser controlada pelo utilizador.
- Pode ser robusto para valores discrepantes (ou seja, o usuário pode especificar uma decomposição robusta), para que observações ocasionais incomuns não afetem as estimativas do ciclo de tendência e dos componentes sazonais. No entanto, eles afetarão o componente restante.

Por outro lado, o STL tem algumas desvantagens, (Hyndman, 2018). Em particular, ele não lida automaticamente com a variação dos dias uteis da semana, e fornece apenas facilidades para decomposições aditivas.

Decomposições entre aditivo e multiplicativo podem ser obtidas usando uma transformação Box-Cox dos dados com $0 < \lambda < 1$. Um valor de $\lambda = 0$, corresponde à decomposição multiplicativa enquanto $\lambda = 1$, é equivalente a uma decomposição aditiva.

3.4 Modelo de suavização exponencial

Os métodos estatísticos de séries temporais são técnicas quantitativas frequentemente utilizadas para realizar prognósticos de variáveis, dentre os quais se encontram os métodos com suavização exponencial.

O ajuste exponencial é uma outra forma de estimar a tendência de uma série temporal. Apresenta algumas vantagens em relação às médias móveis:

- permite realizar previsões de curto prazo (para o período seguinte da série), o que não é possível por médias móveis.
- leva em conta todos os valores previamente observados ao período sob análise, e não somente os "mais próximos" dele, como ocorre nas médias móveis.

Na realidade o ajuste exponencial fornece uma média móvel exponencialmente ponderada ao longo da série temporal, ou seja, cada previsão ou valor ajustado depende de todos os valores prévios. Os pesos designados para os valores observados decrescem ao longo do tempo, ou seja, o valor observado mais recentemente recebe o maior peso, o valor anterior o segundo maior e o valor observado inicialmente recebe o menor peso, imagina-se nesse sentido que os dados mais recentes devem ter mais influência nas previsões do que os mais antigos.

Makridakis et al. (2008), os modelos exponenciais com ajuste assumem a representação matemática dada por :

$$\hat{Y}_{T+1|T} = \alpha Y_T + \alpha(1-\alpha)Y_{T-1} + \alpha(1-\alpha)^2 Y_{T-2} + \cdots , \tag{3.1}$$

A escolha da constante de regularização α é crucial para o ajuste exponencial, mas é um processo subjetivo, dado a existência de vastos factores que influenciam essa escolha. Não obstante, é possível estabelecer uma regra de escolha:

- se o interesse é simplesmente obter a tendência, eliminando o efeito das outras componentes, o valor de α deverá ser próximo de zero;
- se houver interesse, porém, em realizar previsão com a série é recomendável que o valor de α seja mais próximo de 1, de maneira a refletir melhor o comportamento da série no curto prazo.

O tipo mais simples de série temporal é aquele em que os valores da série flutuam aleatoriamente em torno de um valor fixo, sem apresentar qualquer tendência. O método estatístico com suavização exponencial simples para a previsão de séries temporais é apropriado para dados estacionários nos quais não há tendência significativa nos dados no decorrer do tempo.

Este procedimento denominado suavização exponencial simples utiliza uma ponderação distinta para cada valor observado na série temporal, de modo que valores mais recentes recebam pesos maiores. Dessa forma, os pesos formam um conjunto que decai exponencialmente a partir de valores mais recentes (Hyndman & Athanasopoulos, 2018).

3.5 Modelo de Holt - método com suavização exponencial dupla

O método com suavização exponencial dupla é, em geral, um mecanismo de previsão para dados de série temporal que apresentam uma tendência linear.

Desenvolvida por Holt (1957) esse método representa um desenvolvimento da suavização exponencial simples para dados de séries temporais que apresentam tendência linear.

Este método oferece refinamentos adicionais na modelação, à medida que se introduz uma constante de suavização que afeta a tendência da série. A representação da função de previsão do método de Holt é-nos dado por:

$$\text{Equação de previsão : } \hat{Y}_{t+h|t} = \ell_t + hb_t \tag{3.2}$$

$$\text{Equação de nível : } \ell_t = \alpha Y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \tag{3.3}$$

$$\text{Equação de tendência : } b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}, \tag{3.4}$$

Onde ℓ_t representa uma estimativa do nível da série no momento t, b_t representa uma estimativa da tendência (inclinação) da série no momento t, α é o parâmetro de suavização para o nível $0 \leq \alpha \leq 1, \beta^*$ é o parâmetro de suavização da tendência, $0 \leq \beta^* \leq 1$.

Como na suavização exponencial simples, a equação de nível aqui mostra que ℓ_t é uma média ponderada de observação Y_t e a previsão de treinamento um passo à frente para o tempo t, dado por $\ell_{t-1} + b_{t-1}$. A

equação da tendência mostra que b_t é uma média ponderada da tendência estimada no momento t baseado em $\ell_t - \ell_{t-1}$, e b_{t-1} a estimativa anterior da tendência.

3.6 Método com suavização exponencial tripla (método de Holt-Winter)

O método de suavização exponencial tripla é um desenvolvimento do método de previsão de Holt (1957).

Desenvolvido por Winter (1960) para aplicar à série temporal que exibe tendência e sazonalidade, é um dos métodos mais utilizados para previsão de curto prazo, devido a sua simplicidade, custo operacional baixo e boa precisão.

Sabendo que certas séries possuem sazonalidade, nível e tendência que capta características da série que se repetem em intervalos de tempo regulares, Winter propõe métodos de projeção para essas séries, considerando dois tipos de efeitos sazonais: aditivos e multiplicativos.

1. Método de Holt-Winter para efeitos sazonais aditivos

   O método de Holt-Winter para efeitos sazonais aditivos é utilizado quando a amplitude do ciclo sazonal nao depende do nível local da série, isto e, continua constante ao longo do tempo. Seja um modelo cuja série sazonal e formada pela soma de nível, tendência, um fator sazonal e um erro aleatório, com um período p:

   $$\hat{Y}_{t+h|t} = \ell_t + hb_t + s_{t+h-m(k+1)} \tag{3.5}$$

   Nesse método, além da função (3.5) que calcula a previsão, três outras funções são utilizadas para estimar o nível, a tendência da série no período atual e os valores do fator sazonal correspondente ao último período de sazonalidade, conforme equações abaixo.

   $$\ell_t = \alpha(Y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \tag{3.6}$$

   $$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \tag{3.7}$$

   $$s_t = \gamma(Y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}, \tag{3.8}$$

   são as constantes α (parâmetros) de suavização que controlam o peso relativo ao nível ℓ_t, a tendência b_t e a sazonalidade s_t, respectivamente.

2. Método de Holt-Winter para efeitos sazonais multiplicativos

   Uma leve modificação no modelo anterior torna-se apropriado para a modelação de dados de série temporal estacionária com efeitos sazonais multiplicativos. O método de Holt-Winter para efeitos sazonais multiplicativos é utilizado na modelação de dados onde a amplitude do ciclo sazonal varia de forma proporcional ao nível da série ao longo do tempo. Seja um modelo de série sazonal, de período s, fator sazonal multiplicativo e tendência aditiva, isto é, descreve o comportamento estrutural da série, as projeções dos valores futuros da série são efetuadas através da função de previsão do método representada por:

   $$\hat{Y}_{t+h|t} = (\ell_t + hb_t)s_{t+h-m(k+1)} \tag{3.9}$$

onde $\hat{Y}_{t+h|t}$ é a previsão para h períodos à frente $t+h$. Nesse método, além da função (3.9) que calcula a previsão três outras funções são utilizadas para estimar o nível, a tendência da série no período atual e os valores do fator sazonal correspondente ao último período de sazonalidade, conforme equações, respectivamente.

$$\ell_t = \alpha \frac{Y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1}) \tag{3.10}$$

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \tag{3.11}$$

$$s_t = \gamma \frac{Y_t}{(\ell_{t-1} + b_{t-1})} + (1-\gamma)s_{t-m} \tag{3.12}$$

São as constantes (parâmetros) de suavização que controlam o peso relativo ao nível α, a tendência β^* e a sazonalidade γ, respectivamente (Hyndman, 2018).

4

Modelos lineares de séries temporais

Na classe dos modelos paramétricos, a análise é feita no domínio do tempo. Dentre estes modelos os mais frequentemente usados são os modelos baseados na autocorrelação dos dados, os modelos auto-regressivos e de médias moveis (ARMA), os modelos auto-regressivos integrados e de médias móveis (ARIMA), modelos de memória longa (ARFIRMA), modelos estruturais e modelos não-lineares. A função de autocorrelação desempenha um papel importante na análise de modelos paramétricos.

4.1 Processo MA(q)

No processo de médias móveis, tem-se a variável Y_t que sofre a influência da média, e além disso ela vai sofrer a influência dos termos estocásticos, que serão também os termos de ruidos brancos, estes que possuem média zero e variância constante ao longo do tempo.

Nesse sentido Y_t é-nos dado por:

$$Y_t = \alpha + u_t + \theta_1 e_{t-1} \qquad MA(1) \tag{4.1}$$

$$Y_t = \alpha + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \theta_q e_{t-q} + e_t \qquad MA(q) \tag{4.2}$$

$$Y_t = \alpha + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \theta_q e_{t-q} + \theta_{q+n} e_{t-(q+n)} ... \qquad MA(\infty) \tag{4.3}$$

O termo médias moveis MA(q) indica quantas vezes deve-se desfasar o modelo. Nesse sentido, na equação (4.1), o termo de erros ou termo estocástico, foi desfasado 1 vez, enquanto que na equação (4.2), o termo estocástico foi desfasado (q) vezes, para na equação (4.3), demonstrar-se que se pode efetuar em termos matemáticos infinitos desfasamentos, correspondendo nesse caso ao $MA(\infty)$.

Este modelo é então uma combinação linear dos modelos de ruído branco, tendo por caraterística importante as séries serem estacionárias, quando os erros forem ruídos brancos, para ordem de desfasagem (q) conhecida;

4.2 Processo AR(p)

Um processo auto-regressivo AR(p), parâmetro $p \in \mathbb{N}$, pode-se representar por:

$$Y_t = \mu + \phi_{t-1} + e_t \qquad AR(1) \tag{4.4}$$

$$Y_t = \mu + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + e_t \qquad AR(p) \tag{4.5}$$

$$Y_t = \mu + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + e_t \qquad AR(\infty) \tag{4.6}$$

Sendo que a ordem do modelo depende do número de defasagem da variável Y_t, e todos os termos do erro sejam do tipo ruído branco de média zero.

Diferente do modelo de média móvel, que pode-se verificar facilmente se o modelo é ou não estacionário, o processo auto-regressivo não é claramente possível devido ao processo de passeio aleatório. Seja $Y_t = \phi Y_{t-1} + e_t, \quad se \quad \phi = 1$, então estaremos num processo de passeio aleatório.

Considerando: $\mu = 0$

$$Y_t = \phi Y_{t-1} + e_t \tag{4.7}$$

$$Y_t = \phi(\phi Y_{t-2} + e_{t-1} + e_t)$$

$$Y_t = \phi^2 Y_{t-2} + \phi e_{t-1} + e_t$$

$$Y_t = \phi^2(\phi Y_{t-3} + e_{t-2} + \phi e_{t-1} + e_t)$$

$$Y_t = \phi^3 Y_{t-3} + \phi^2 e_{t-2} + \phi e_{t-1} + e_t ...$$

$$Y_t = \sum_{n=1}^{\infty} \phi^k e_{t-k} \tag{4.8}$$

Ou seja, o AR(1) é sobre certas condições equivalente ao modelo MA(∞) com o $\phi_i = \theta^i$, através do processo de recursividade.

Para estabilizar o MA(ϕ), temos que $\sum \mid \phi^i \mid < \infty$, como a sequência de ϕ_i é uma progressão geométrica, para ser estacionário é somente necessário que o $|\phi| < 1$, tendo nesse caso uma soma dos termos infinitos de uma progressão geométrica, com os termos convergente se $\mid \phi \mid < 1$, logo a série será estacionária. No caso dos passeios aleatórios será igual a 1, passando a ser não estacionário.

4.3 Processo ARMA (p, q)

O modelo ARMA e composto pela soma dos modelo MA e o modelo AR, sendo então

$$Y_t = \theta_1 u_{t-1} + \theta_2 u_{t-2} + ... + \theta_p u_{t-p} + \mu + e_t + \phi_1 y_{t-1} + ... + \phi_q y_{t-q} \tag{4.9}$$

O uso desse modelo é de uso limitado devido a questão de estacionariedade, para resolver esse problema utiliza-se o mecanismo de transformação (diferenciação e/ou transformação de Box Cox) transformando o modelo ARIMA em um modelo ARMA.

4.4 Processo ARIMA (p, d, q)

A criação do modelo ARIMA tem como metodologia de análise das etapas a metodologia desenvolvida por Box-Jenkins. Essa metodologia utiliza séries temporais estacionárias e não estacionárias, sendo modeladas para fazer previsão.

O modelo ARIMA é-nos dado por:

$$Y_t = \theta z_{t-1} + \theta z_{t-2} + ...\theta_p z_{t-p} + \mu + e_t + \phi_1 e_{t-1} + ... + \phi_q e_{t-q} \tag{4.10}$$

4.5 Processo SARIMA(p,d,q)(P,D,Q)$_s$

Em determinados tipos de dados, podemos verificar que existem uma certa periodicidade de período fixo. Para modelar essa periodicidade utiliza-se o modelo sazonal autoregressivo integrado de médias móveis (SARIMA).

Esse modelo trata séries não estacionárias com periodicidade fixa, com picos e declínios ao longo da sua evolução temporal.

Box & Jenkins (1976) propõem o modelo sazonal multiplicativo, onde a ordem é dada por SARIMA $(p, d, q)x(P, D, Q)_s$, onde s é o período da série.

$$\phi(B)\Phi(B^s)(1-B)^d(1-B^s)^D y_t = \theta(B)\Theta(B^s)e_t, \tag{4.11}$$

em que,

- $\phi(B) = 1 - \phi_1 B^1 - ... - \phi_p B^p$ é o polinómio autorregressivo de ordem p
- $\Phi(B^s) = 1 - \Phi_1 B^s - ... - \Phi_p B^{Ps}$ polinómio autorregressivo sazonais de ordem P
- $(1 - B^s)^D$ é o operador diferença generalizada, quando duas observações estao distantes entre si de s intervalos de tempos que apresentam alguma semelhança, e D é o numero de diferenças de defasagem s necessárias para retirar a sazonalidade da série.
- $(1 - B)^d$ é o operador diferença e d é o número de diferenças necessárias para retirar a tendencia da série;
- $\theta(B) = 1 - \theta_1 B^1 - ... - \theta_q B^q$ é o polinómio de médias moveis de ordem q;
- $\Theta(B^s) = 1 - \theta_s B^s - ... - \theta_Q B^{Qs}$ é o polinómio de médias moveis sazonal de ordem Q;

É importante salientar que as séries com a componente sazonal podem ser diferenciadas tanto na sua componente não sazonal, quanto na seu componente sazonal, ou em ambos.

4.6 Etapas de modelação

A construção do modelo Box & Jenkins se basea em um ciclo iterativo, no qual a escolha da estrutura do modelo é baseada nos próprios dados.

a) Especificação – Uma classe geral de modelos é considerada para a análise. Tal escolha pode ser feita, por exemplo, usando a FAC e a FACP estimadas, que esperamos que representem adequadamente as respectivas teóricas, que são desconhecidas;

b) Identificação - consiste na criação de um modelo com base na análise das funções de autocorrelações e autocorrelações parciais. O procedimento de identificação para determinar os valores de p, d e q do modelo ARIMA(p, d, q) envolve: verificar se precisa de transformação não linear (como a transformação Box-Cox, que estabiliza a variância), e tomar diferenças para deixar a série estacionária (valor d) e encontrar o ARMA (p, q);

Todavia, a FAC de processos AR e ARMA podem apresentar um comportamento complicado e, por consequência, a utilização de tal ferramenta não parece muito adequada para a identificação de tais processos. Box, Jenkins e Reinsel (1994) propõem o uso da função de auto-correlação parcial (FACP), que é útil para identificar modelos AR.

c) Estimação dos parâmetros do modelo identificado - as estimativas preliminares encontradas na fase de identificação serão usadas como valores iniciais nesse procedimento;

Após termos identificado um modelo provisório para a série temporal, o passo seguinte é estimar seus parâmetros. O método de estimação mais utilizado e o da máxima verossimilhança.

d) Validação do modelo ajustado - tendo todos os parâmetros sido significativos, deve-se passar a análise de resíduos, que consiste em efetuar o teste à correlação e à heterocedasticidade (quando o modelo de hipótese matemático apresenta variâncias não iguais para todas as observações) dos resíduos, para saber se este é adequado para os fins de ajustamento e/ou previsao. Caso o modelo não seja adequado, o ciclo é repetido.

Depois da estimação dos parâmetros do modelo encontrado, deve-se verificar se ele representa, ou não, adequadamente, os dados. Tal verificação pode ser feita através de uma análise de resíduos, pois, a partir dos mesmos, teremos ideia se as suposições feitas sobre o modelo são ou não válidas. Ainda, veremos, também, que qualquer insuficiência revelada pode sugerir um modelo alternativo como sendo adequado.

Se o modelo proposto for adequado, os resíduos deverão assemelhar-se a um ruído branco e, portanto, deverão ter média zero, variância constante e ser aproximadamente não autocorrelacionados. Uma ferramenta bastante útil para se verificar a suposição de que os erros são um ruído branco se baseia no cálculo da função de auto-correlação (FAC) dos resíduos.

Muitas vezes os usuários da metodologia Box & Jenkins identificam não apenas um único modelo, mas alguns modelos que serão então estimados e verificados. Assim, se o propósito for o de fazer previsões, escolher-se-á entre os modelos ajustados o melhor, por exemplo, no sentido de fornecer o menor erro quadrático médio de previsão.

5

Redes neuronais

As redes neuronais artificiais são inspiradas pela maneira como o cérebro humano funciona, sendo este o órgão mais importante do corpo humano. O cérebro humano processa todas as funções desempenhadas pelo ser humano, podendo processar grandes quantidades de informações através dos dados recebidos dos sentidos humanos, sobretudo a visão. O processamento é feito por células nervosas, que atuam nos sinais elétricos que passam por eles e aplicam uma lógica, para a transmissão do sinal.

Uma célula nervosa é definida como uma célula que transmite impulsos nervosos ou sinais eletroquímicos. Sempre foi desafiador entender o funcionamento do cérebro humano; contudo, com os avanços em tecnologias de computação, agora pode-se artificialmente programar redes neuronais biológicas.

A figura 5.1 mostra a estrutura de uma célula nervosa biológica:

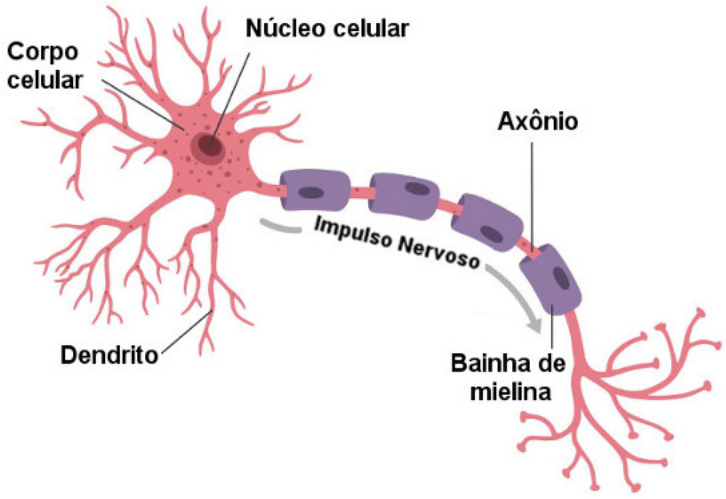

Figura 5.1: Neurónio biológico
Fonte: mundoeducacao.uol.com.br/biologia/neuronios.htm

Os principais componentes de cada célula nervosa são:

Dendrites: pontos de entrada em cada neurónio que recebem a entrada de outros neurónios na rede na forma de impulsos elétricos.

Corpo da celula: gera inferências a partir das entradas das dendrites e decide que ação tomar.

Terminais do axónio: Eles transmitem saídas na forma de impulsos elétricos para o próximo neurónio.

5.1 Redes neuronais artificiais

Semelhante à estrutura biológica da célula nervosa, as redes neuronais artificiais (RNAs) definem o neurónio como uma unidade de processamento central, que realiza uma operação matemática para gerar uma saída a partir de um conjunto de entradas. A saída de um neurónio é uma função da soma ponderada das entradas mais o viés, (Venkateswaran & Ciabuschi, 2017).

O modelo proposto por Rosenblatt, conhecido como *perceptron*, era composto por uma estrutura de rede, tendo como unidades básicas, nós, e por uma regra de aprendizagem. Anos depois Rosenblatt demonstrou o teorema de convergência do *perceptron*, demonstrou a convergência para questões linearmente separáveis, se um nó for treinado com o algoritmo de aprendizagem do *perceptron*.

As redes perceptron contém uma camada de nós de saída, conectados às entradas por conjuntos de pesos. A soma do produto dos pesos pelas entradas é calculada por cada nó de saída e, se o valor calculado ultrapassar um certo limiar, o neurónio dispara e ajusta a saída para um determinado valor; caso contrário, a saída é ajustada para outro determinado valor. Podem existir mais de uma camada *layer* de neurónios, que podem interagir entre si ou não.

Nas redes neuronais unidirecionais (fig. 5.2), os sinais percorrem somente um caminho: da entrada para a saída. Não há *feedback*, isto é, a saída de qualquer camada não afeta a mesma camada.

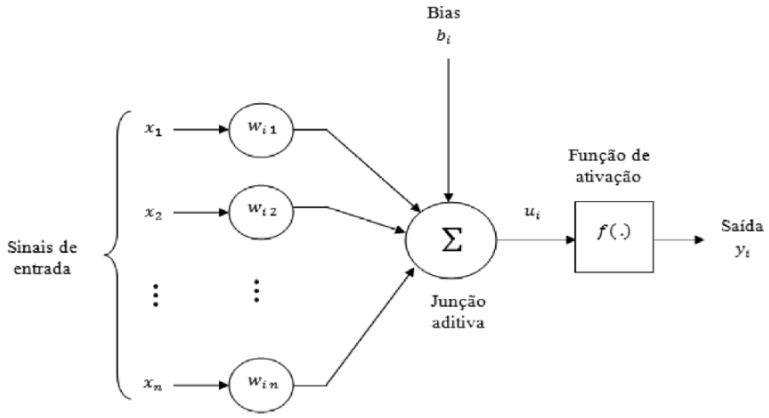

Figura 5.2: Redes neuronais unidirecionais.
Fonte: https://https://www.scielo.br/scielo.php?script=sci$_artextpid = S2179 - 84512019000200229$

Já as redes neurais multidirecionais, representada na figura 5.3, conhecidas como redes neuronais recorrentes (RNN) podem ter sinais indo em ambas as direções, introduzindo *loops* na rede. Os processamentos advindos da entrada anterior são realimentadas na rede, o que lhes concede uma certa memória. Seu "estado" está se modificando continuamente até um ponto de equilíbrio, permanecendo nesse ponto de equilibrio até que novas entradas mudem esse "estado"e um novo equilíbrio precise ser encontrado.

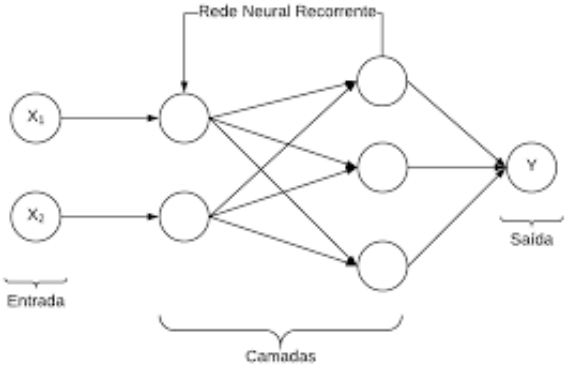

Figura 5.3: Redes neuronais multidirecionais.
Fonte: https://www.google.com/

5.2 História

Os primeiros trabalhos sobre as redes neuronais começaram com o realizado por McCullock e Pitts (1943), com o objetivo de modelar matematicamente o comportamento de um neurónio biológico. Esse trabalho inicial se concentrava sobretudo em descrever um modelo artificial de um único neurónio e apresentar suas capacidades computacionais ao em vez de apresentar técnicas de aprendizado.

"O aprendizado de redes biológicas e artificiais veio a ser objeto de estudo somente alguns anos depois do trabalho de McCulloch e Pitts. O primeiro trabalho de que se tem notícia que tem ligação direta com o aprendizado foi apresentado por Donald Hebb em 1949" (Braga, 2000). Hebb mostrou que modificando os pesos de entrada dos nós, é possível o aprendizado devido à plasticidade da aprendizagem de redes neuronais.

Hebb propôs que a conectividade do cérebro é continuamente modificada conforme um organismo vai aprendendo tarefas funcionais diferentes e que agrupamentos neuronais são criados por tais modificações. Mais tarde, Widrow e Hoff (1960), sugeriram uma regra de aprendizado, conhecida como, regra delta, que é ainda hoje bastante utilizada.

"A regra delta é baseada no método do gradiente para minimização do erro na saída de um neurónio com resposta linear" (Simon Haykin,1999).

Em 1958, Rank Rosenblatt, demonstrou, com o seu novo modelo, o *perceptron*, que, se fossem acrescentadas de sinapses ajustáveis, as RNAs com nos MCP poderiam ser treinadas para classificar certos tipos de padrões. O perceptron simples descrito por Rosenblatt possui três camadas: a primeira recebe as entradas do exterior e possui conexões fixas; a segunda recebe impulsos da primeira através de conexões cuja eficiência de transmissão é ajustável e, por sua vez, envia saídas para a terceira camada. Inicialmente, a saída da rede é aleatória, mas, pelo ajuste gradual dos pesos, o perceptron é treinado para fornecer saídas de acordo com os dados do conjunto de treinamento.

Em 1969, Minsky e Papert chamaram a atenção para algumas tarefas que o perceptron não era capaz de executar, já que este só resolve problemas linearmente separáveis, ou seja, problemas cuja solução pode ser obtida dividindo-se o espaço de entrada em duas regiões através de uma reta (Braga, 2000).

Nos anos 70. a abordagem conexionista ficou adormecida em grande parte devido à repercussão do trabalho de Minsk e Papert apesar de alguns poucos pesquisadores continuarem trabalhando na área.

Em 1982 John Hopfield publicou um artigo que chamou a atenção para as propriedades associativas das RNAs, possibilitando a retomada das pesquisas na área. O grande feito de Hopfield foi sem dúvida mostrar a relação entre redes recorrentes auto-associativas e sistemas físicos. Não obstante, a descrição do algoritmo de treinamento *back-propagation* alguns anos mais tarde mostrou que a visão de Minsky e Papert sobre o perceptron era bastante pessimista.

A partir de meados dos anos 80 houve um reavivamento do interesse pelas RNAs na comunidade internacional. Dois outros fatores foram responsáveis pela retomada de interesse na área: em primeiro lugar, o avanço da tecnologia, em segundo, o fato de a escola simbolista, a despeito do seu sucesso na solução de determinados tipos de problemas, não ter conseguido avanços significativos na resolução de alguns problemas simples para um ser humano (Braga, 2000).

As redes neuronais percorreram um longo caminho desde os trabalhos de McCulloch e Pitts, e hoje, elas estabeleceram-se como um assunto interdisciplinar com raízes profundas em neurociências, psicologia, matemática, economia entre outras áreas (Simon Haykin, 1999).

5.3 Redes *perceptron* múltiplas camadas

O trabalho original de McCulloch e Pitts teve como foco a modelação de um neurónio biológico e sua capacidade computacional com a apresentação de vários exemplos de topologias de rede com capacidade de execução de funções booleanas. Na altura surgiam os primeiros computadores digitais, podendo-o associar ao pensamento da época, de que era possivel construir a partir de operadores lógicos basicos, computadores intelligentes. Mas foi somente em 1958, com o trabalho de Frank Rosenblatt que o conceito de aprendizado em redes neuronais artificiais foi introduzido (Braga, 2000).

Quando as camadas do perceptron são combinadas, elas formam uma arquitetura multicamadas, e isso dá a complexidade necessária do processamento da rede neuronal de perceptrons de múltiplas camadas (MLPs) e são a arquitetura mais amplamente usada hoje para redes neurais (Balaji, 2017).

5.4 Função de ativação

A abstração do processamento de redes neuronais é conseguida principalmente por meio das funções de ativação. Uma função de ativação é uma função matemática que converte a entrada numa saída e adiciona a magia do processamento da rede neural. Sem funções de ativação, o funcionamento das redes neurais será como funções lineares. Uma função linear é aquela em que a saída é diretamente proporcional à entrada (Ciaburro & Balaji, 2017).

No entanto, a maioria dos problemas que as redes neurais tentam resolver são não lineares e de natureza complexa. Para atingir a não linearidade, as funções de ativação são usadas.

Os problemas mundiais não são lineares portanto, as funções de ativação contribuem para um mapeamento linear, independentemente do aprendizado. as funções de ativação são usadas nas camadas ocultas das redes neurais e são separadas apenas com base no seu alcance ou grau de linearidade (Ghatak, 2019).

"A partir do modelo proposto por McCulloch e Pitts, foram derivados vários outros modelos que permitem a produção de uma saída qualquer, não necessariamente zero ou um. Essa possibilidade se deve a diferentes funções de ativação" (Braga, 2000).

As funções de ativação podem ser linear e não lineares.

1. Função de ativação linear

   A função de ativação linear $f(x) = ax + b$ é uma linha reta (linear) e a saída dessas funções varia de $]-\infty, +\infty[$. Devido a complexidade redes os modelos lineares tem grande dificuldade de utilização. Com as funções de ativação linear, podemos colocar quantas camadas ocultas na rede neural, e a saída final ainda será uma combinação linear dos dados de entrada (Ghatak, 2019).

2. Função de Ativação Não Linear

- Função logistica

A razão pela qual se usa a função sigmóide é porque ela varia entre (0 a 1). Portanto, é especialmente usado para modelos em que temos que prever a probabilidade como uma saída. Como a probabilidade de qualquer coisa existe apenas entre 0 e 1, sigmóide é a escolha certa. A função é diferenciável. Isso significa que podemos encontrar a inclinação da curva sigmóide em quaisquer dois pontos.

A função sigmóide logística pode fazer com que uma rede neural fique presa no momento do treinamento.

A função logistica é representada por: $f(x) = \frac{1}{1+e^x}$ cujo o gráfico se representa na figura 5.4:

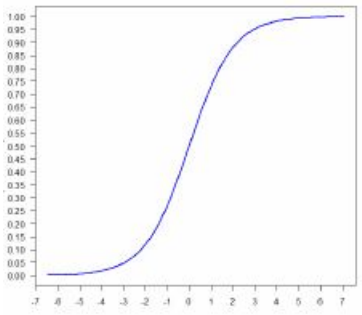

Figura 5.4: Função sigmóide
Fonte: https://pt.wikipedia.org/wiki/Logistica

- Função ReLU

As redes com a função ReLU são fáceis de otimizar, já que a ReLU é extremamente parecida com a função identidade, mas produzindo zero em metade do seu domínio. Como consequência, as derivadas se mantêm grandes enquanto a unidade estiver ativa (Facure, 2017).

A função Relu é representada por: $f(x) = max(0, x)$

Cujo o gráfico se representa na figura 5.5:

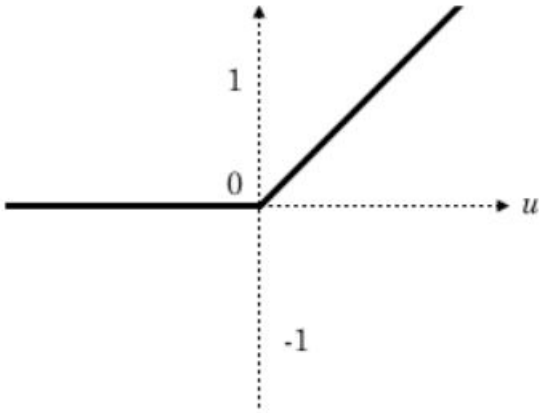

Figura 5.5: Função ReLU
Fonte: https://pt.wikipedia.org/wiki/Funcaoafim

A ativação ReLU sendo mais eficiente do que as funções sigmoidais vistas acima contribuiu de uma forma significativa para a recente avanço e popularidade de *Deep Learning* (que terá um aprofundamento mais adiante). Essa não linearidade é um ótimo exemplo de como a simplicidade pode ser extremamente poderosa.

- Função Tangente Hiperbólica (TanH)

Tal como a função sigmóide, a função Tangente Hiperbólica (TanH) também tem um formato em 'S', mas varia entre [-1,1], em vez [0,1] como na função sigmoide. A TanH se aproxima mais da identidade, sendo assim uma alternativa mais atraente do que a sigmoide para servir de ativação às camadas ocultas das RNAs (Facure, 2017).

A função tanH representa-se por: $y = tgh(x)$

A representação gráfica da função tangente hiperbólica segue na figura 5.6:

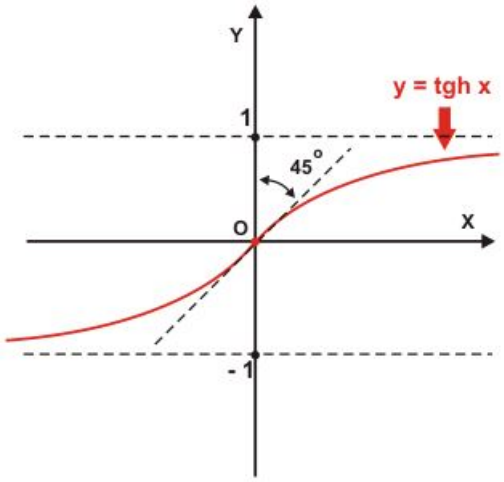

Figura 5.6: Função Tangente Hiperbólica
Fonte:https://www.alfaconnection.pro.br/matematica/funcoes/funcoes-hiperbolicas/tangente-hiperbolica/

6

Machine Learning

A aprendizagem de máquina consiste em treinar um modelo ou algoritmo com dados e, em seguida, usar o modelo para prever quaisquer novos dados. As máquinas podem ser ensinadas como o ser humano a fazer uma tarefa com base no treinamento. Primeiro, fornecemos dados suficientes para dizer à máquina o que precisa ser feito em determinadas circunstâncias. Após o treinamento, a máquina pode funcionar automaticamente e também aprender a se ajustar. Esse tipo de treinamento da máquina é denominado aprendizado de máquina.

"Campo de estudo que dá aos computadores a capacidade de aprender sem serem explicitamente programados".

Arthur Samuel

6.1 Dados de treino e dados de teste

No processo de aprendizagem existe uma divisão dos dados de treino e dados de teste.

A 1ª etapa, que é o processo de treinamento consiste em calibrar os pesos de cada nó de modo iterativo, partindo de valores aleatórios (normalmente entre 0 e 1). A taxa de aprendizagem (também um valor entre 0 e 1) diz o quão rápido a rede chega ao seu processo de classificação: um valor muito pequeno causa demora a convergir para o real valor, enquanto que um valor muito alto pode levar para valores fora do ajuste e nunca convergir.

Existem dois tipos de treino de máquina, o treinos supervisionado onde os cálculos efetuados com o padrão de entrada e os respetivos pesos resultarem no padrão de saída informado dentro de uma faixa considerada satisfatória e o treinamento não supervisionado, onde os neurónios usam somente valores de entrada, tentando classificá-los mediante algum critério de semelhança.

A 2ª etapa é o teste, que consiste na validação dos parâmetros do modelo.

6.2 Sobreajuste e Subajuste

O sobreajuste se refere a um modelo que foi treinado demais nas informações dos dados de treino. Um modelo que está super ajustado não terá bom desempenho em dados novos. O sobreajuste é sem dúvida o problema mais comum no aprendizado de máquina e é especialmente problemático porque um modelo que parece ser altamente preciso terá um desempenho mau.

Segundo Abhijit Ghatak (2019) o ajuste excessivo ocorre quando o modelo é altamente complexo, resultando na captura do ruído presente nos dados, em vez de capturar a tendência nos dados.

Na figura 6.1 tem-se o caso de um sobreajuste.

Figura 6.1: Representação de um sobreajuste
Fonte:https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/

O subajuste normalmente se refere a um modelo que não foi treinado o suficiente. Isso pode ser devido ao tempo de treinamento incorreto ou não foi treinado adequadamente. Um modelo que não está muito bem ajustado terá um mau desempenho nos dados de treinamento, bem como nos dados novos.

Segundo Abhijit Ghatak(2019) a falta de ajuste ocorre quando existe um viés alto no modelo e, portanto, não é possível capturar a tendência nos dados.

Na figura 6.2 temos o caso de um subajuste.

Figura 6.2: Representação de um subajuste
Fonte:https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/

O estado ideal é chamado de generalização. É aqui que o modelo tem bom desempenho nos dados de treinamento e nos novos dados não vistos durante o processo de treinamento.

Na figura 6.3 temos o caso de um ajuste esperado.

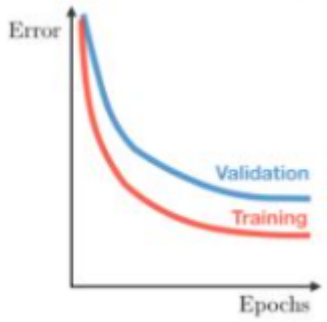

Figura 6.3: Representação de um ajustamento esperado
Fonte:https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/

6.3 Aprendizado supervisionado e aprendizado não supervisionado

- O **treinamento supervisionado**, carateriza-se pelo conhecimento apriori das entradas e das saídas. Os erros são propagados e os pesos ajustados automaticamente. No entanto, algumas redes nunca aprendem e não convergem se não houver dados suficientes para permitir a aprendizagem completa. Idealmente, deve haver dados suficientes para uma divisão dos dados entre treino e teste e o conjunto de treino deve ser grande o suficiente para conter todas as informações para aprender relacionamentos que são importantes (Darpa, 1992).

  Se uma rede simplesmente não consegue resolver o problema, deve-se rever as entradas e saídas, o número de camadas, o número de elementos por camada, as conexões entre as camadas, a função de transferência e até mesmo os próprios pesos iniciais. Existem vários algoritmos usados para ajustar os pesos durante o treinamento, e o mais utilizado, é o da retropropagação.

  Para evitar o esquecimento é conveniente treinar a rede com todos os dados disponíveis. Os fatos anteriores podem ser esquecidos em aprender algo novo. Como resultado, o sistema precisa aprender tudo junto, encontrando as melhores configurações de peso para o conjunto total de dados. Cabe

encontrar o melhor formato de dados e arquitetura de rede correspondente para um determinado problema.

- O outro tipo de treinamento é chamado de **não supervisionado**. No treinamento não supervisionado, à rede são fornecidas entradas, mas não saídas desejadas. O próprio sistema deve então decidir quais recursos usará para agrupar os dados de entrada. Essas redes não usam influências externas para ajustar os pesos. Estes redes procuram regularidades ou tendências nos sinais de entrada e fazem adaptações de acordo com a função da rede (Darpa, 1992).

  Atualmente, a aprendizagem não supervisionada não é bem compreendida. Essa adaptação ao meio ambiente é a promessa que permitiria à ciência tipos de robôs de ficção que aprendem continuamente por conta própria à medida que encontram novas situações e novos ambientes. Estamos sempre em situações onde conjuntos de treinamento exatos não existem, por causa desse aspeto continua a haver pesquisas e esperanças nesse campo.

6.4 Pré-processamento

A utilização de dados de séries temporais no campo da machine learning, requer um preparamento dos dados de forma a adequar os dados aos algoritmos apropriados de forma a ter uma melhor *performance* possível tanto no ajustamento dos dados como na previsão. Existem várias técnicas necessárias para efetuar essa preparação citada abaixo:

Estruturação dos dados Essa etapa necessária e fundamental consiste na preparação da série temporal e colocá-la em formato de aprendizado supervisionado de máquina. Essa preparação consiste em colocar a série em formato matricial, onde constam os dados de entrada e os dados de saída.

$$A = \begin{bmatrix} x_{11} & 0 & 0 & 0 & 0 \\ x_{21} & x_{11} & 0 & 0 & 0 \\ x_{31} & x_{21} & x_{11} & 0 & 0 \\ x_{41} & x_{31} & x_{21} & x_{11} & 0 \\ x_{51} & x_{41} & x_{31} & x_{21} & x_{11} \\ x_{n1} & x_{n1} & x_{n1} & x_{n1} & x_{n1} \end{bmatrix}$$

Dados de treino um conjunto de treino é usado para ensinar a rede a compreender os padrões da série (e não a série por si mesmo), onde a recomendação que se situa entre os 70% e os 80%. No entanto, qualquer que seja a proporção selecionada, deve-se prestar atenção para garantir que o conjunto de dados de treinamento seja suficientemente grande para conseguir captar os padrões e grande não demasiado para não apreender os padrões.

Dados de teste o conjunto de dados restante pode em seguida ser usado para testar a rede treinada nos dados de treino. Infelizmente, ainda não existe nenhum guia de seleção disponível para dividir o conjunto de dados preparado em dois subconjuntos.

Normalização dos dados consiste num ajustamento dos dados na forma dos algoritmos de leitura de máquina, ou seja, consiste numa preparação dos dados para o treinamento. Inclui a normalização de dados originais pré-processados para o intervalo operacional da rede, de modo que os dados normalizados sejam estritamente moldados para atender aos requisitos da camada de entrada de rede e são adaptados às não linearidades dos neurónios (Palit, 2005). A normalização consiste em fazer com que todas as observações variem entre [0,1], onde se consegue aplicando a fórmula abaixo.

$$x_{ni} = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

7

Fundamentos do *deep learning*

Inicialmente o aprendizado de máquina se concentrava no aprendizado de apenas uma ou duas camadas de representações dos dados de entrada. Isso provou ser intratável para resolver problemas de perceção humana. Por isso, deu lugar a uma nova visão sobre representações de aprendizado, que enfatizava o aprendizado de múltiplas camadas sucessivas de representações, resultando no conceito de aprendizagem profunda. Uma camada é uma função de transformação de dados que realiza a transformação dos dados que passam por essa camada. Essas transformações são parametrizadas por um conjunto de pesos e viés, que determinam o comportamento da transformação nessa camada.

Na figura 7.1 temos o caso de uma rede neuronal artificial profunda, que se caracteriza pela existência de várias camadas ocultas.

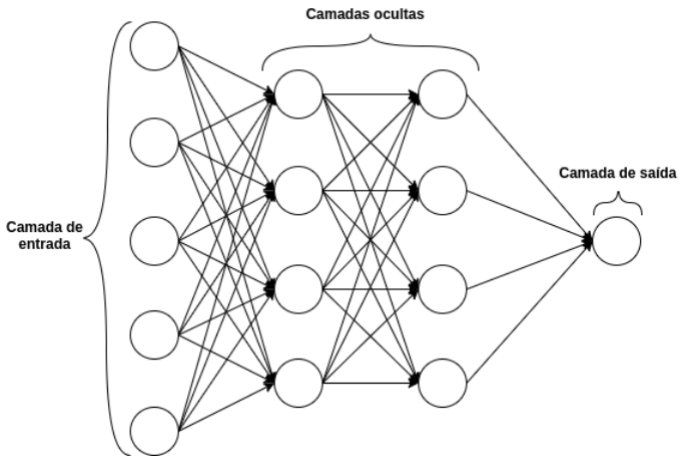

Figura 7.1: Redes neuronais profundas
Fonte:https://www.google.com/aprendizagem profunda

7.1 Otimização

Existem algumas escolhas importantes e algumas vezes subtis que precisamos fazer no processo de construção e treinamento de uma rede neuronal. Precisamos decidir qual a função de ativação, quantas camadas deve-se usar, quantos neurónios devemos ter nas camadas, qual algoritmo de otimização é mais adequado para a nossa rede, como inicializamos os nossos pesos de parâmetro, escolha do parâmetro de regularização, escolha da taxa de aprendizado, o tempo de treinamento acima de um parâmetro. Tudo isso no sentido de otimizar o desempenho do algoritmo.

Muitos fatores contribuem para o desempenho de um modelo. A maneira como medimos o desempenho, como pode ser óbvio para alguns, é por uma função de custo.

Função de custo

A função de custo nos fornece um valor que queremos otimizar. Existem muitas funções de custo, mas uma das funções de custo mais simples e frequentemente usadas é a soma das diferenças quadráticas.

$$C = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{7.1}$$

Onde y_i é o que queremos que a saída seja e $\hat{y}_i$ sendo a saída prevista real de uma rede neuronal. Basicamente, para cada amostra n, começamos a somar a partir do primeiro exemplo $i = 1$ e sobre todos os quadrados das diferenças entre a saída que queremos y_i e a saída prevista $\hat{y}_i$ para cada observação.

7.1.1 O mecanismo das redes neuronais: otimização baseada em gradiente descendente

A descida do gradiente é um algoritmo de otimização usado para encontrar os valores de pesos e desvios do modelo com o objetivo de minimizar a função de perda.

Depois de encontrar a magnitude e a direção do gradiente, podemos reduzir o peso por um hiperparâmetro (são parâmetros ajustáveis que devem ser ajustados para obter um modelo com desempenho ideal) conhecido como taxa de aprendizado e atualize o valor do novo peso (na direção dos mínimos).

No método descendente do gradiente o treinamento se processa em lotes (que correspondem ao conjunto total da série) ou minilotes (que correspondem a uma parte da série) onde os parâmetros são atualizados com base no atual e continuam iterativamente. No geral o lote que contém os dados desejados é referido como uma época.

Na figura 7.2 temos a representação do gradiente.

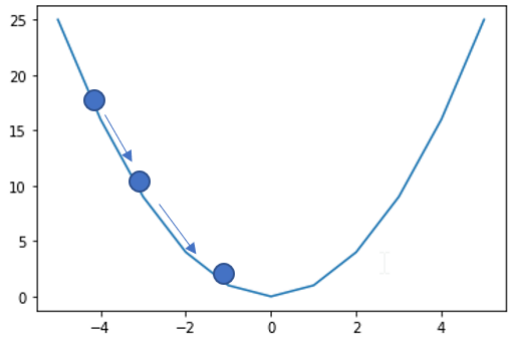

Figura 7.2: Otimização baseada em Gradiente descendente
Fonte:https://docs.aws.amazon.com/pt_br

7.1.2 Descida do gradiente estocástico

Na descida do gradiente estocástico, pega-se um minilote de amostra aleatória e realizamos uma atualização dos pesos e desvios com base no gradiente médio do minilote. Os pesos para cada minilote são inicializados aleatoriamente num valor pequeno, entre [0,1].

Quando o tamanho do minilote é definido como um, se realiza uma atualização num único exemplo de treinamento, e este é um caso especial da descida do gradiente de minilote conhecida como descida do gradiente estocástico. É chamado de "estocástico" porque escolhe aleatoriamente um único exemplo para executar uma atualização até que todos os exemplos no conjunto de dados de treinamento tenham sido vistos. O ideal é que, em qualquer um dos algoritmos otimizadores, o nosso objetivo seja o mínimo global da função, que representará os melhores possíveis valores de parâmetro. Dependendo de onde começamos no espaço de parâmetros, é provável que possamos encontrar mínimos locais e pontos de sela ao longo do caminho. Os pontos de sela são um caso especial de mínimos locais em que as derivadas nas direções ortogonais são zero.

Dizemos que queremos alcançar um mínimo global, o ponto mais baixo da função. Porém, isso nem sempre é possível. É muito provável que atinjam um mínimo local, que é um ponto entre a inclinação que se move

para cima, do lado esquerdo e do lado direito. Se encontrarmos um mínimo, dizemos que a nossa rede neuronal convergiu. Se não, ou vemos uma queda estranha no desempenho, dizemos que a rede neuronal divergiu.

7.1.3 Derivadas de encadeamento: o algoritmo de retropropagação

No processo de treinamento de redes perceptron de múltiplas camadas, é aplicado o algoritmo de aprendizado de retropropagação do erro, conhecido também como regra delta generalizada. A regra delta ajusta iterativamente os pesos da rede para minimizar as diferenças entre a saída atual e a saída desejada.

Esse ajuste é calculado através da minimização do erro quadrático pelo método do gradiente descendente.

A retropropagação serve para calcular os gradientes com eficiência, sendo este o coração de toda rede neuronal, existindo um outro conceito que são os otimizadores servindo este para treinar a rede neuronal usando os gradientes calculados com a retropropagação. Em suma, tudo o que a retropropagação faz é calcular os gradientes.

Só podemos alterar os pesos e o viés, mas as ativações são cálculos diretos desses pesos e viés, o que significa que indiretamente podemos ajustar todas as partes da rede neural para obter a saída desejada, exceto a camada de entrada, pois esse é o conjunto de dados que se insere.

Obviamente, existem muitos fatores que contribuem para o desempenho de uma determinada rede neuronal, complexidade do modelo, hiperparâmetros (taxa de aprendizado, funções de ativação, etc.), tamanho do conjunto de dados e muitos outros fatores.

7.2 Treinamento de redes *perceptron* múltiplas camadas

Como referido anterormente o algoritmo de aprendizado mais conhecido para treinamento destas redes é o algoritmo de retropropagacao. A maioria dos métodos de aprendizado para redes neuronais artificiais (RNAs) do tipo percetron de múltiplas camadas (MLP) utiliza variações deste algoritmo.

O algoritmo de retropropagação é um algoritmo supervisionado que utiliza pares (entrada, saída desejada) para, por meio de um mecanismo de correção de erros, ajustar os pesos da rede. O treinamento ocorre em duas fases, em que cada fase percorre a rede em um sentido. Estas duas fases são chamadas de fase *forward* e fase *backward*. A fase *forward* é utilizada para definir a saída da rede para um dado padrão de entrada. Ilustra-se na figura 7.3:

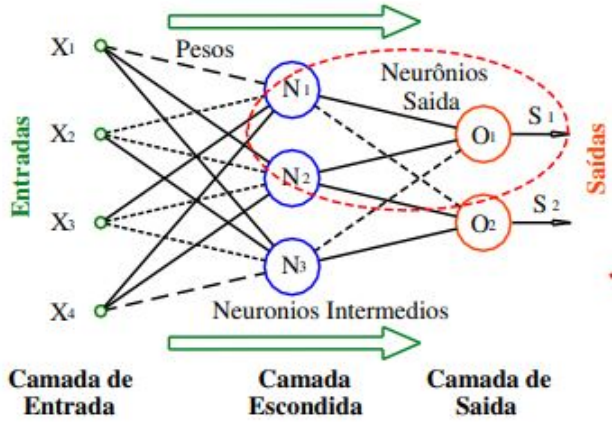

Figura 7.3: Treinamento *forward*
Fonte: PUC - Rio

A fase *backward* faz uso da saída desejada e fornecida pela rede para atualizar os pesos de suas conexões. Ilustra-se na figura 7.4:

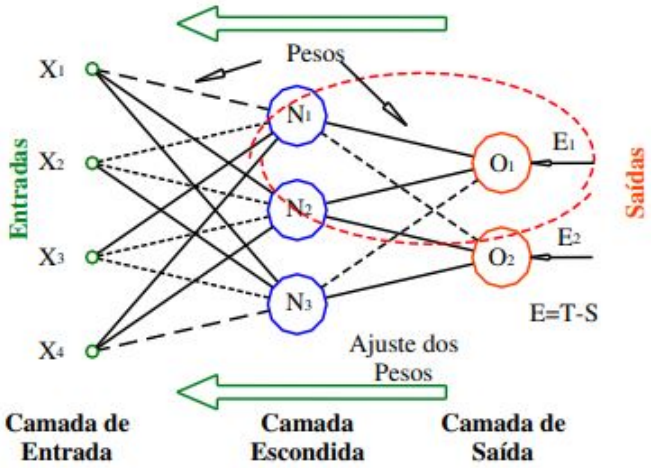

Figura 7.4: Treinamento *backward*
Fonte: PUC - Rio

A fase *backward* envolve as etapas citadas a seguir:
1. Começa da última camada, até chegar a primeira camada (de entrada);
2. Os nós da camada atual ajustam seus pesos de forma a reduzir seus erros;
3. O erro de um nó das camadas intermediárias é calculado utilizando os erros dos nós da camada seguinte conectados a ele, ponderados pelos pesos das conexões entre eles.

O algoritmo back-propagation procura minimizar o erro obtido pela rede ajustando pesos e limiares para que eles correspondam às coordenadas dos pontos mais baixos da superfície de erro. Para isto, ele utiliza um método de gradiente descendente.

Um dos problemas enfrentados no treinamento de redes MLP diz respeito à definição de seus parâmetros. Pequenas diferenças nestes parâmetros podem levar a grandes diferenças tanto no tempo de treinamento quanto na generalização obtida. Não é raro encontrar, para um mesmo problema, utilizando o mesmo

método de treinamento, tempos de treinamento que diferem em uma ou mais ordens de magnitude.

Naturalmente surge o problema de quando parar de treinar a rede. Existem vários métodos para a determinação do momento em que o treinamento deve ser encerrado. Estes métodos designam-se por critérios de paragem. Os critérios de paragem mais utilizados são:

1. encerrar o treinamento após N ciclos;

2. encerrar o treinamento após o erro quadrático médio ficar abaixo de uma constante a;

3. encerrar o treinamento quando a percentagem de classificações corretas estiver acima de uma constante a (mais indicado para saídas binárias);

4. combinação dos métodos acima descritos.

Outro aspecto que precisa ser observado é a frequência de atualização dos pesos. A freqüência de ajuste dos pesos influencia o desempenho obtido durante o treinamento. Duas abordagens diferentes têm sido utilizadas quanto à freqüência (periodicidade) para o ajuste de pesos pelo algoritmo backpropagation:

- por padrão (on-line);

- por ciclo (batch).

Em cada uma destas abordagens existem vantagens e desvantagens. Na abordagem por padrão, os pesos são atualizados após a apresentação de cada padrão de treinamento. Esta abordagem é estável se a taxa de aprendizado for pequena (é aconselhável reduzir progressivamente esta taxa). A rede se torna instável se se utilizar uma taxa muito elevada. A abordagem por padrão é geralmente mais rápida, sobretudo se o conjunto de treinamento for grande e redundante. Uma outra vantagem desta técnica é que ela requer menos memória.

Numa execução por ciclo, os pesos são atualizados após todos os padrões terem sido apresentados. Esta técnica pode ser lenta caso o conjunto de treinamento for grande, apesar de ser geralmente mais estável. Esta abordagem apresenta uma estimativa mais precisa do vetor gradiente, à custa da necessidade de mais memória. A escolha da abordagem a ser utilizada depende da aplicação e da distribuição estatística dos dados.

8

Redes neuronais artificiais de memória longa

Com inspiração nas redes neuronais biológicas, os neurónios artificias têm uma estrutura que é uma característica importante na definição do tipo de arquitetura das redes neuronais artificias (RNAs) a utilizar. Diferente das redes *feed forward*, as redes recorrentes incluem um *loop de feedback* através do qual a saída do passo $n-1$ é alimentada novamente à rede para afetar o resultado do passo n, e assim sucessivamente para cada etapa subsequente.

O fluxo dos sinais em redes neurais pode ser em apenas uma direção ou em recorrência. No primeiro caso, chamamos a arquitetura de rede neural *feed-forward* , uma vez que os sinais de entrada são alimentados na camada de entrada e, depois de processados, são encaminhados para a próxima camada (Balaji, 2017).

Quando a rede neuronal tem algum tipo de recorrência interna, o que significa que os sinais são realimentado para um neurónio ou camada que já recebeu e processou esse sinal, a rede é do tipo *feedback* (Balaji, 2017).

A razão especial para adicionar recorrência numa rede é a produção de um comportamento dinâmico, particularmente quando a rede aborda problemas envolvendo séries temporais ou padrões de reconhecimento, que requerem uma memória interna para reforçar o processo de aprendizagem. Contudo, tais redes são particularmente difíceis de treinar, eventualmente deixando de aprender. A maioria das redes de *feedback* são de camada única, mas é possível construir uma rede multicamadas recorrente, como redes MLP recorrentes (Balaji, 2017).

8.0.1 Funcionalidade

Numa rede multicamadas, o resultado de cada nó é obtêm-se pela combinação dos resultados realizados pelos nós das camadas anteriores (ou pode-se dizer que o resultado de cada nó resulta do processo das camadas anteriores a ele e que estão conectados a este. Quando se segue da primeira camada intermediária em direção à camada de saída, as funções implementadas se tornam cada vez mais complexas. Estas funções definem como é realizada a divisão do espaço de decisão. Para uma rede com pelo menos duas camadas intermediárias, pode-se dizer que o seguinte processamento ocorre em cada uma das camadas:

- Primeira camada intermediária: cada nó traça retas no espaço de padrões de treinamento.

- Segunda camada intermediária: cada nó combina as retas traçadas pelos neurónios da camada anterior conectado a ele

- Camada de saída: cada nó forma regiões que são combinações das regiões definidas pelos nós a ele conectados da camada anterior.

Pode-se dizer que as camadas intermédias de uma rede *perceptron* de múltiplas camadas captam o padrão dos dados, gerando códigos internos dos padrões de entrada, utilizado posteriormente para a saída da rede. Se existe um número suficientemente grande de camadas intermediárias, é possível formar representações internas para qualquer conjunto de padrões de entrada.

8.1 Rede LSTM - Rede de memória de longo prazo

A rede de memória de longo prazo (LSTM) foi introduzida por Hochreiter & Schmidhuber em 1997, e é treinada usando *backpropagation through time* (BPTT). O grande problema das redes neuronais recorrentes é a perda de informação quando têm-se longos períodos temporais, e essa rede veio colmatar esse problema (que é problema do gradiente de fuga), recordar a informação por longos períodos de tempo é praticamente o seu comportamento padrão. Os neurónios das redes (LSTM) conseguem isso mantendo uma célula exclusiva para o armazenamento e fluxo da memória, além de portões do tipo entrada, saída e esquecimento, que controlam esse fluxo. A rede de memória de longo prazo (LSTM) têm aplicações poderosas em dados sequenciais, e podem criar redes grandes e recorrentes para lidar com problemas no aprendizado de máquina para sequências difíceis (Filippo, 2017).

As LSTMs são consideradas evoluções da RNNs. Quando introduzimos uma memória de longo prazo no RNN, somos capazes de lembrar muito informações anteriores e usá-las para o processamento atual. Este conceito é denominado LSTM, modelo de RNN, que tem vários casos de uso em vídeo, áudio, previsão de texto e várias outras aplicações (Balaji, 2017).

A célula LSTM é composta por 5 componentes não lineares diferentes, interagindo um com o outro de uma

maneira particular. O estado interno de uma célula é modificado apenas pelo LSTM por meio de interações lineares. Isso permite que as informações sejam propagadas sem problemas ao longo do tempo, com um consequente aumento da capacidade de memória da célula. LSTM protege e controla as informações na célula através de três portas, que são implementados por uma sigmóide e uma multiplicação pontual. Para controlar o comportamento de cada porta, um conjunto de parâmetros são treinados com gradiente descendente, a fim de resolver um determinado problema.

Desde a sua definição inicial (Hochreiter e Schmidhuber 1997), diversas variantes da unidade LSTM original foram propostas na literatura. Na sequência, refiro à arquitetura normalmente utilizada proposta por Graves e Schmidhuber (2005). As equações de diferença que definem o passe para frente para atualizar o estado da célula e para calcular a saída estão listados abaixo

$$\text{Portão de esquecimento: } \sigma_f[t] = \sigma(\ \mathbf{W}_f\mathbf{x}[t] + \mathbf{R}_f y[t-] + \mathbf{b}_f),$$

$$\text{Estado candidato: } \widetilde{y}[t] = g_1(\mathbf{W}_h\mathbf{x}[t] + \mathbf{R}_h y[t-1] + \mathbf{b}_h),$$

$$\text{Porta de entrada : } \sigma_u[t] = \sigma(\mathbf{W}_u\mathbf{x}[t] + \mathbf{R}_u y[t-1] + \mathbf{b}_u),$$

$$\text{Estado da célula: } \mathbf{h}[\text{t}] = \sigma_u[t] \odot \widetilde{y}[t] + \sigma_f[t] \odot \mathbf{h}[t-1],$$

$$\text{Porta de saida : } \sigma_0[t] = \sigma(\mathbf{W}_0\mathbf{x}[t] + \mathbf{R}_0 y[t-1] + \mathbf{b}_0),$$

$$\text{Saida : y[t]} = \sigma[t] \odot g_2(\mathbf{h}[t]).$$

$\mathbf{x}[t]$ é o vetor de entrada no tempo t. $\mathbf{W}_f, \mathbf{W}_h, \mathbf{W}_u$ e $\mathbf{W}_o$ são pesos que são aplicadas à entrada da célula LSTM. $\mathbf{R}_f, \mathbf{R}_h, \mathbf{R}_u$ e $\mathbf{R}_o$ são matrizes quadradas que definem os pesos das conexões recorrentes, enquanto $\mathbf{b}_f, \mathbf{b}_h, \mathbf{b}_u$ e $\mathbf{b}_o$ são vetores de polarização. A função $\sigma(.)$ é um sigmóide, enquanto $g_1(.)$ e $g_2(.)$ são funções de ativação não linear pontuais geralmente implementadas como tangentes hiperbólicas que comprimam os valores entre $[-1, 1]$. Finalmente,$\odot$, é a multiplicação de entrada entre dois vetores.

Cada porta na célula tem uma funcionalidade específica e única. O portão de esquecimento σ_f decide quais informações devem ser descartadas do estado de célula anterior $\mathbf{h}[t-1]$. A porta de entrada σ_u opera no estado anterior $\mathbf{h}[t-1]$, após ter sido modificada pela porta de esquecimento, e decide quanto o novo estado $\mathbf{h}[t]$ deve ser atualizado com um novo candidato $\mathbf{h}[t]$. Para produzir a saída $\mathbf{y}[t]$, primeiro a célula filtra sua corrente estado com uma não linearidade $g_2(.)$. Então, a porta de saída σ_0 seleciona a parte do estado para ser retornado como saída. Cada porta depende da entrada externa atual $\mathbf{x}[t]$ e do células anteriores produzem $\mathbf{y}[t-1]$.

Na figura 8.1 temos o esquema da rede LSTM.

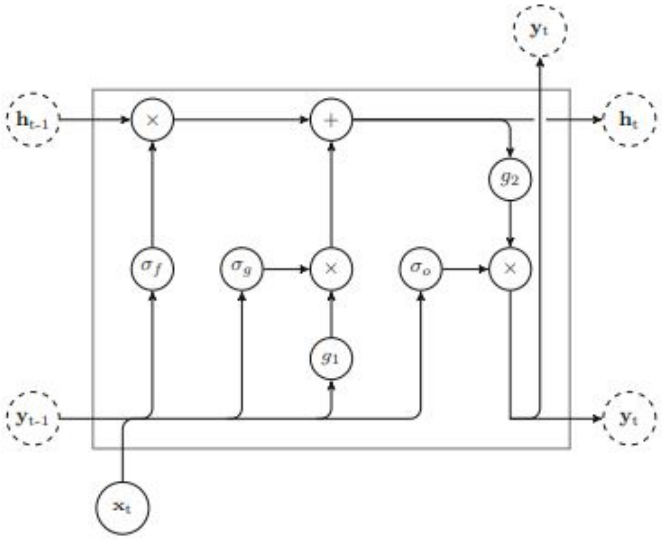

Figura 8.1: Rede de memória de Longo Prazo (LSTM)
Fonte: Bianchi et al., Recurrent Neural Networks for Short-Term Load Forecasting, 2017

8.2 Rede unidades recorrentes com portão (GRU)

As redes neuronais com unidades recorrentes com portão (GRU) são uma variação das LSTMs. O portão de entrada e saída é substituído por um portão de atualização, que é responsável por controlar quanto de informação reter e quanto atualizar. No lugar do portão de esquecimento temos um portão de *reset*, que funciona de forma análoga, mas cuja localização na estrutura é diferente. As GRUs costumam ter aplicação similar ás das LSTMs, mas são mais rápidas e fáceis de treinar. Entretanto, o seu desempenho pode ser um pouco menos expressivo (Filippo, 2017).

A Gated Recurrent Unit (GRU) é outra famosa arquitetura fechada, originalmente proposto por Cho et al. (2014), que captura de forma adaptativa dependências em diferentes prazos. No GRU, as portas de esquecimento e entrada são combinadas numa única porta de atualização que controla de forma adaptativa quanto cada unidade oculta pode lembrar ou esquecer. O estado interno em GRU está sempre totalmente exposto na saída, devido à falta de um controle mecânico, como a porta de saída em LSTM (Filippo, 2017).

GRUs foram testados pela primeira vez por Cho et al. (2014). Em uma comparação empírica de GRU e LSTM, configurada com a mesma quantidade de parâmetros, Chung et al. (2014) concluiu que em alguns conjuntos de dados, o GRU pode superar o LSTM, tanto em termos de capacidade de generalização quanto em termos de tempo necessário para alcançar a convergência e atualizar parâmetros (Balaji, 2017).

Uma representação da célula GRU é relatada na Figura 8.2. GRU faz uso de dois portões. O primeiro é a porta de atualização, que controla quanto o conteúdo atual da célula deve ser atualizado com o novo estado candidato. O segundo é o portão *reset* que, se fechado (valor próximo a 0), pode efetivamente redefinir a memória da célula e faça a unidade agir como se a próxima entrada processada fosse a primeira na sequência. As equações de estado da GRU são as seguintes:

$$\text{Portão de Redefinição : } \mathbf{r}[t] = \sigma(\mathbf{W}_r\mathbf{h}[t-1] + \mathbf{R}_r\mathbf{x}[t] + \mathbf{b}_r),$$

$$\text{Estado atual: } \mathbf{h'}[t] = \mathbf{h}[t\text{-}1] \odot \mathbf{r}[t],$$

$$\text{Estado candidato : } \mathbf{z}[t] = g(\mathbf{W}_z\mathbf{h'}[t-1] + \mathbf{R}_z\mathbf{x}[t] + \mathbf{b}_z),$$

$$\text{Portão de atualização: } \mathbf{u}[t] = \sigma(\mathbf{W}_u\mathbf{h}[t-1] + \mathbf{R}_u\mathbf{x}[t] + \mathbf{b}_u),$$

$$\text{Novo estado: } \mathbf{h}[t] = (1 - \mathbf{u}[t] \odot \mathbf{h}[t-1] + \mathbf{u}[t] \odot \mathbf{z}[t])$$

Aqui, $g(.)$ é uma função não linear geralmente implementada por uma tangente hiperbólica. Em uma célula GRU, o número de parâmetros é maior do que na unidade ERNN, mas menor do que em uma célula LSTM. Os parâmetros a serem aprendidos são as matrizes retangulares $\mathbf{W}_r, \mathbf{W}_z, \mathbf{W}_u$, as matrizes quadradas $\mathbf{R}_r, \mathbf{R}_z, \mathbf{R}_u$ e os vetores de polarização $\mathbf{b}_r, \mathbf{b}_z, \mathbf{b}_u$.

Na figura 8.2 temos o esquema da rede GRU.

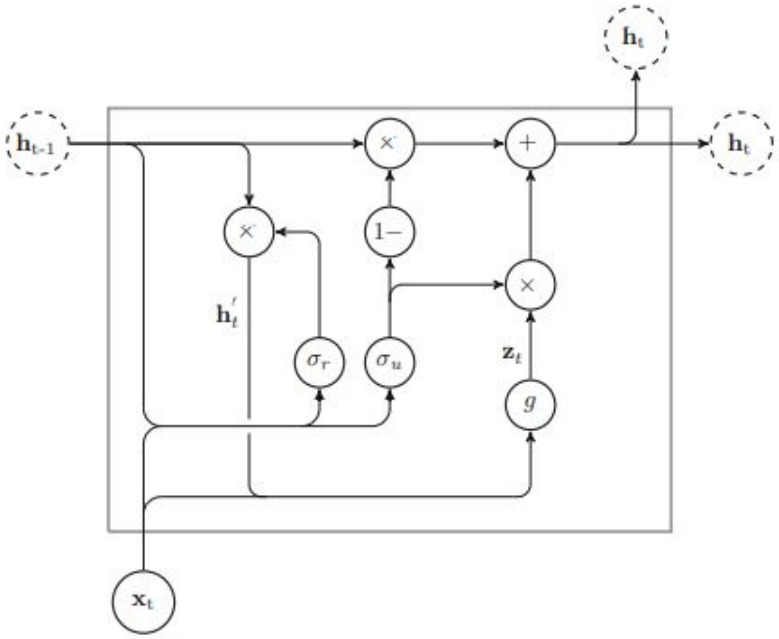

Figura 8.2: Rede de unidades recorrentes com portão (GRU)
Fonte: Bianchi et al., Recurrent Neural Networks for Short-Term Load Forecasting, 2017

9

Modelação de séries temporais: modelos tradicionais versus redes neuronais

9.1 Séries temporais com tendência

Todas as análises foram feitas no *software* R Cran (versão 4.0.0) e considerou nos testes estatísticos um nível de significância de 5%.

9.1.1 Consumo público em Portugal do primeiro trimestre do ano de 1995 ao quarto trimestre do ano de 2012

Os dados da série consumo público ilustrado, iniciam no primeiro trimestre do ano de 1995, vai até ao quarto trimestre do ano de 2012. Graficamente verifica que a série possui uma tendência crescente, para no início do ano 2010 ter uma quebra, e retomando o crescimento em meados de 2012.

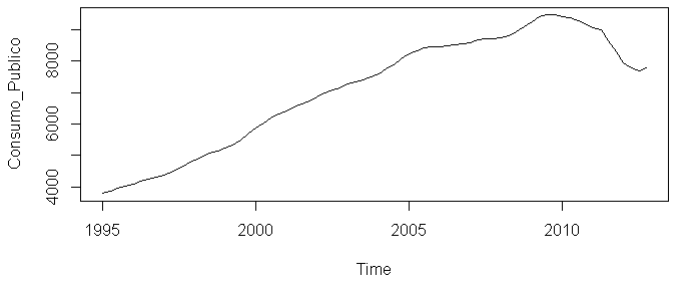

Figura 9.1: Série consumo público em Portugal entre o primeiro trimestre do ano de 1995 e o quarto trimestre do ano de 2012

Foram realizados os testes de Mann-Kendall e de Cox-Stuart para testar a existência de tendência, sendo que no teste de Mann-Kendall as hipóteses é que não existe tendência (hipótese nula) *versus* existe tendência (hipótese alternativa) e no teste de Cox-Stuart a hipótese nula é a mesma mas na hipótese alternativa pode especificar-se se a tendência é crescente ou decrescente.

Obteve-se um $P - value < 2.2e - 16$ para o teste de Mann-Kendall e um $P - value = 1.455e - 11$ para o teste de Cox-Stuart, donde existe evidência para afirmar que a série é não estacionária com tendência crescente. Tendo em conta esta evidência efetuou-se uma transformação de Box-Cox ($\tilde{\lambda} = 0.82$), seguida de uma diferenciação simples. As funções de autocorrelação (FAC) e autocorrelação parcial (FACP) seguem nos gráficos:

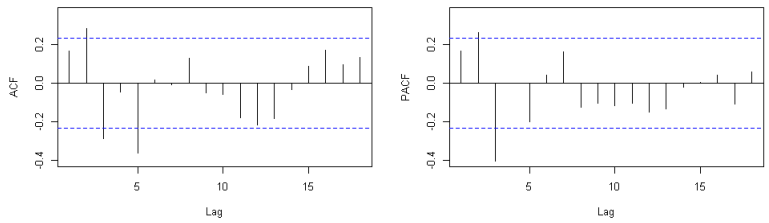

Figura 9.2: FAC e FACP da série transformada do consumo público em Portugal

Analizando as funções anteriores, verificamos que para a série do consumo público (que possui tendência crescente) o melhor modelo encontrado foi o modelo linear ARIMA(3,2,0), cujos resíduos do modelo ajustado comportam-se como um ruído branco, apresentado na figura 9.3, seguindo uma distribuição normal onde temos o teste de normalidade Kolmogorov-Smirnov ($P - value = 0.06$), sendo que as hipóteses é que seguem uma distribuição normal (hipótese nula) versus não seguem uma distribuição normal (hipótese alternativa), com a média zero, onde o ($P - value = 0.8057$) para o teste t, sendo as hipóteses é que o verdadeiro valor é igual a zero (hipótese nula) versus o verdadeiro valor não é igual a zero (hipótese alternativa).

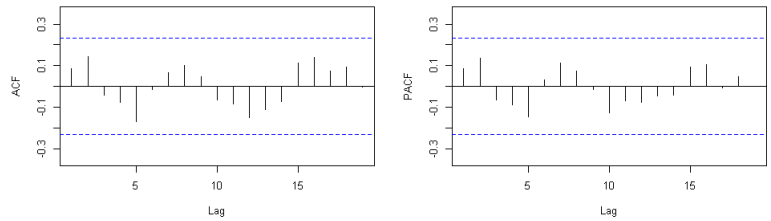

Figura 9.3: Resíduos do modelo auto-regressivo integrado de médias moveis (ARIMA) da série consumo público em Portugal

Quanto ao ajustamento encontrou-se um MAPE = 0.36, ou seja, o modelo teve uma performance de ajustamento de 99.64%, ilustrado na figura 9.4.

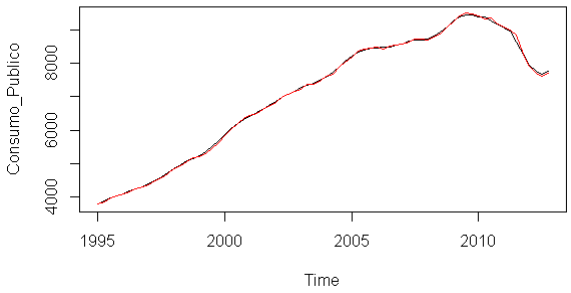

Figura 9.4: Ajustamento utilizando o modelo auto-regressivo integrado de médias móveis (ARIMA) da série consumo público em Portugal

Para obter os resultados do ajustamento utilizando as redes neuronais artificiais e o algoritmo das redes neuronais recorrentes (RNN), utilizou-se para o treinamento os hiperparâmetros apresentados na tabela 9.1, cuja função de activação foi uma logística, com uma taxa de aprendizagem de 0.8, para um treinamento com 90 retropropagações, utilizando um único lote, onde se obteve um erro médio por época igual a 0.044, representado na figura 9.5 abaixo.

Hyperparâmetros	**Resultados**
network type	RNN
sigmoid	logística
learningrate	0.8
hidden-dim	c(4,5)
numepochs	90
batch size	1
update-rule	sgd
use-bias	False
seq-to-seq-unsync	False

Tabela 9.1: Hiperparâmetros do modelo da rede neuronais recorrente da série consumo público em Portugal utilizando o algoritmo RNN

Como pode-se verificar o erro do modelo vai diminuindo a medida que se vai aumentando o número de treinamento por época, mas observa-se a partir da vigésima época de treinamento, pequenas oscilações do

erro a medida que se vai diminuindo, é de observar que o ideal é que aconteçam descidas constantes sem oscilações dos erros no treinamento.

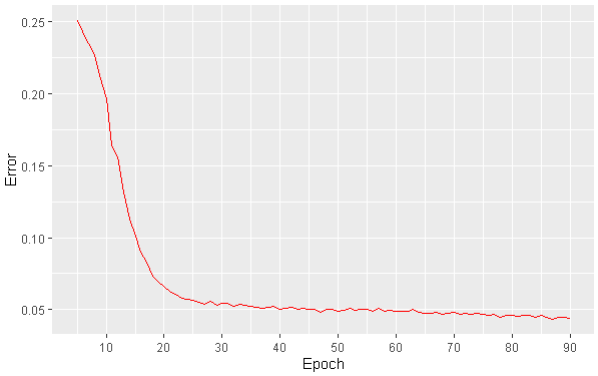

Figura 9.5: Evolução do erro por época utilizando a rede neuronal recorrente da variável consumo público em Portugal.

Quanto ao ajustamento verificamos que a rede vem aprendendo e a melhorar o ajustamento à medida que o erro vai diminuindo. O modelo das redes neuronais recorrentes utilizando o algoritmo RNN obteve uma acurácia de ajustamento MAPE $= 3.20$, ou seja, através do aprendizado de máquina com o algoritmo RNN, o modelo criado tem um acerto no ajustamento de 96.8%. A figura 9.6 nos ilustra essa evolução no aprendizado supervisionado.

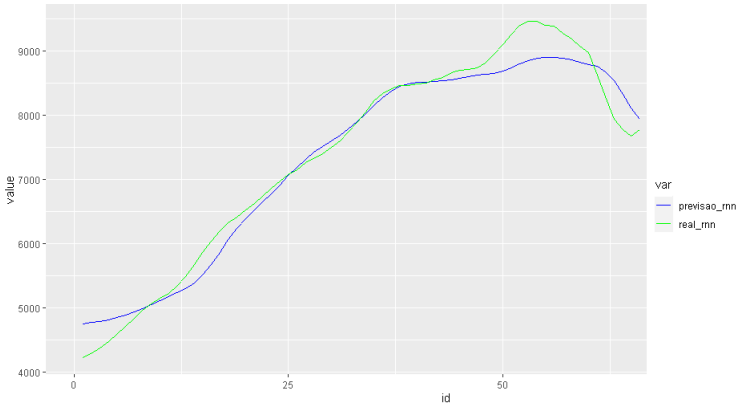

Figura 9.6: Ajustamento utilizando a rede neuronal recorrente com o algoritmo RNN da série consumo público em Portugal

Utilizando o algoritmo das redes unidades recorrentes com portão (GRU), com a função de ativação logística, com uma taxa de aprendizado de 0.8, com um treinamento com 110 retroprogações e um único lote, resumidos na tabela 9.2 verificou-se também que assim como o algoritmo RNN, o erro vai diminuindo com o aumento do número de épocas de treinamento, mas também com algumas oscilações durante todo o treinamento, como mostra a figura 9.7.

Hiperparametros	Resultados
network type	GRU
sigmoid	logistic
learningrate	0.8
hidden-dim	17
numepochs	110
update-rule	sgd
batch size	1

Tabela 9.2: Hiperparâmetros do modelo da rede GRU da série consumo público em Portugal

Como pode-se verificar o erro do modelo vai diminuindo a medida que se vai aumentando o número de treinamento por época, sendo o erro médio por época de 0.08, mas observam-se pequenas oscilações do erro a medida que se vai diminuindo.

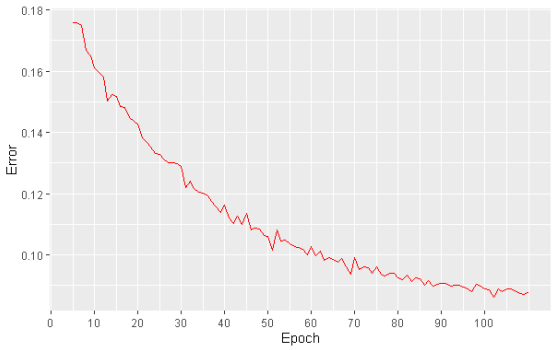

Figura 9.7: Evolução do erro por época utilizando o algoritmo das redes neuronais recorrente (GRU) para a variável consumo público em Portugal

Quanto ao ajustamento constata-se que inicialmente a rede teve um baixo ajustamento, mas a melhorar o ajustamento até que a série original e a série ajustada se sobrepõem. O modelo das redes neuronais recorrentes utilizando o algoritmo GRU obteve um ajustamento com um MAPE = 6.5, ou seja, através do aprendizado de máquina com o algoritmo GRU, o modelo criado tem um acerto no ajustamento de 93.5%. A figura 9.8 nos ilustra essa evolução no ajustamento utilizando o aprendizado supervisionado através do algoritmo GRU.

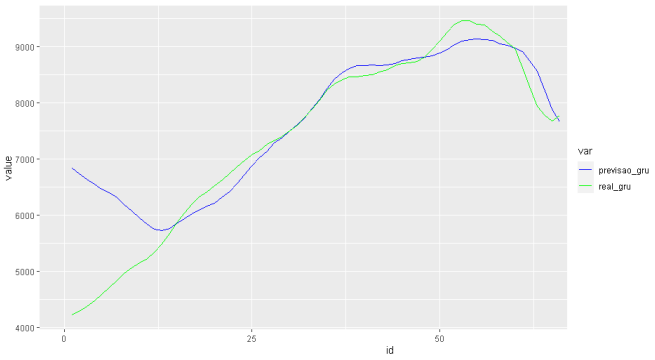

Figura 9.8: Ajustamento da série consumo público em Portugal utilizando o algoritmo GRU

Para obter os resultados do ajustamento utilizando as redes neuronais artificiais e o algoritmo das rede de memória de Longo Prazo (LSTM), utilizou-se para o treinamento os hiperparâmetros apresentados na tabela 9.3, cuja função de activação foi a logística, com uma taxa de aprendizagem de 0.8, para um treinamento com 120 retropropagação, utilizando um único lote, onde se obteve um erro médio por época igual a 0.081.

Hiperparametros	**Resultados**
network type	LSTM
sigmoid	logistic
learningrate	0.8
hidden-dim	c(12)
numepochs	120
batch size	1

Tabela 9.3: Hiperparâmetros do modelo da rede de memória de Longo Prazo (LSTM) da série consumo público em Portugal

Como pode-se verificar na figura 9.9 o erro do modelo utilizando os hiperparametros supracitados para o algoritmo LSTM vai diminuindo a medida que se vai aumentando o número de treinamento por época. Diferente dos algoritmos RNN e GRU, o erro do algoritmo LSTM, tem descidas suaves e contínuas, com menores oscilações ao longo de todo o treinamento.

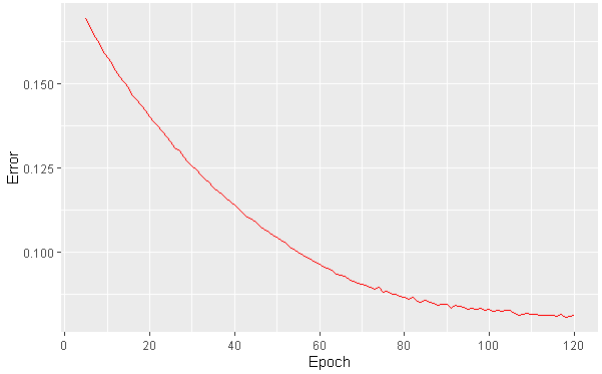

Figura 9.9: Evolução do erro por época utilizando o algoritmo LSTM da variável consumo público em Portugal

Quanto ao ajustamento verificamos que a rede vem aprendendo e a melhorar o ajustamento a medida que o erro vai diminuindo. O modelo das redes neuronais recorrentes utilizando o algoritmo LSTM obteve uma acurácia de ajustamento com um MAPE $= 6.776$, ou seja, através do aprendizado de máquina com o algoritmo LSTM, o modelo criado tem um acerto no ajustamento de 93.23%. A figura 9.10 nos ilustra essa evolução no aprendizado supervisionado com o algoritmo LSTM.

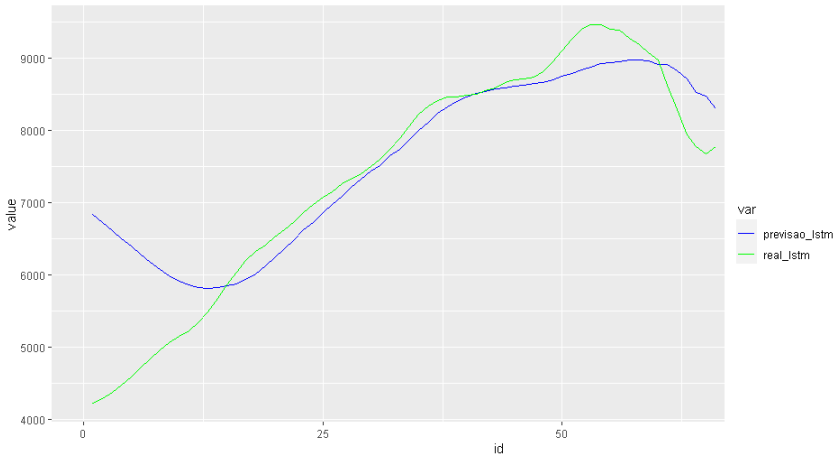

Figura 9.10: Ajustamento da série consumo público em Portugal utilizando o algoritmo LSTM

Efetuando um resumo da *performance* dos diferentes métodos de ajustamento utilizado constata-se um melhor desempenho do modelo linear, seguido da rede RNN, onde apresentou um erro de ajustamento inferior a 5%. Enquanto que as redes GRU e LSTM apresentaram um erro de ajustamento superior a 6%. Esses resultados nos apresenta as primeiras indicações que as redes GRU e LSTM são projetados para séries longas enquanto que a rede RNN foi projetada para séries curtas, demonstrado no desempenho das redes RNN muito próximo dos modelos lineares.

	ME	RMSE	MAE	MPE	MAPE
ARIMA	-8.3267	42.0437	28.1459	-0.15794	0.3586
RNN	31.6493	303.6636	229.3357	0.6553	3.20
GRU	273.735	777.122	444.814	4.2909	6.487
LSTM	215.115	731.679	422.386	3.5359	6.1323

Tabela 9.4: Resumo dos resultados dos diferentes métodos para a série Consumo Público em Portugal

9.1.2 Produto interno bruto em Portugal do primeiro trimestre de 1995 ao quarto trimestre de 2012

Os dados da série Produto Interno Bruto Português (PIB), ilustrados iniciam no primeiro trimestre de 1995, até ao quarto trimestre de 2012. Graficamente verifica que a série possui uma tendência crescente, para no inicio do ano 2010 tiver uma ligeira descida.

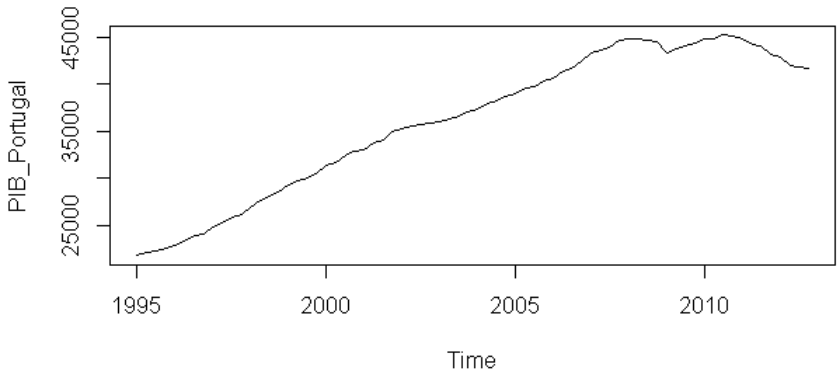

Figura 9.11: Série produto interno bruto de portugal entre o primeiro trimestre de 1995, vai até ao quarto trimestre de 2012

Verifica-se através dos testes de tendência de Mann-Kendall ($P - value < 2.2e - 16$) e de Cox-Stuart ($P - value = 1.455e - 11$) que a série do PIB possui tendência crescente. Deste modo, efetuou-se uma transformação de Box-Cox ($\tilde{\lambda} = 0.523$) e foram feitas duas diferenciações simples.

As funções de autocorrelação e autocorrelação parcial são apresentados na figura 9.12

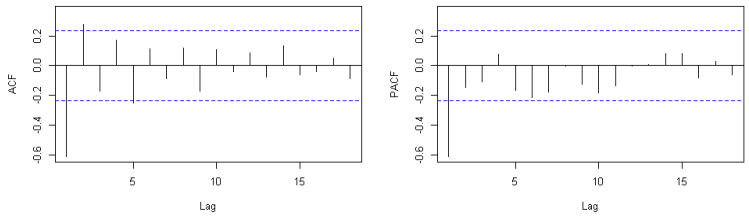

Figura 9.12: Função de autocorrelação e de autocorrelação parcial da série transformada do produto interno bruto de Portugal

Nesse sentido o melhor modelo encontrado foi um ARIMA(1,2,0) cujos resíduos do modelo ajustado comportam-se como um ruído branco, apresentado na figura 9.13, seguindo uma distribuição normal onde temos o teste de normalidade de Kolmogorov-Smirnov($P - value = 0.964$), com a média zero, com um ($P - value = 0.7116$) para o teste t.

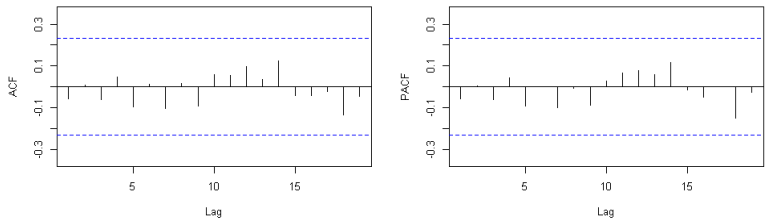

Figura 9.13: Resíduos da série produto interno bruto de Portugal utilizando o modelo auto-regressivo integrado de médias móveis (ARIMA)

Quanto ao ajustamento do modelo encontrado obteve-se um MAPE $= 0.6$, ou seja, o modelo teve uma performance de ajustamento de 99.4% demonstrado na figura 9.14.

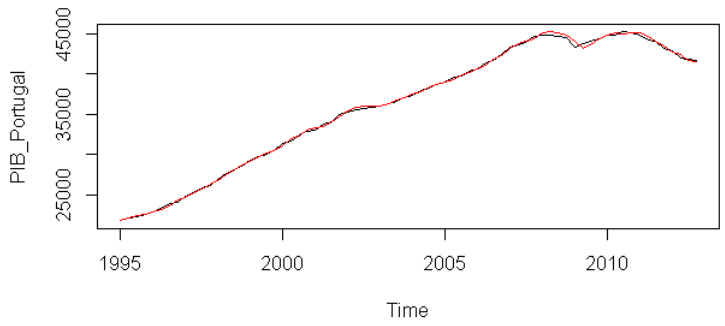

Figura 9.14: Ajustamento da série produto interno bruto de portugal utilizando o modelo auto-regressivo integrado de médias moveis

Para obter os resultados do ajustamento utilizando as redes neuronais artificiais e o algoritmo RNN, utilizou-se para o treinamento os hiperparâmetros apresentados na tabela 9.5, cujo a função de activação foi logística, com uma taxa de aprendizagem de 0.8, para um treinamento com 10 retropropagações, utilizando um único lote, onde se obteve um erro médio por época igual a 0.029, representado na figura **??** abaixo.

Hiperparâmetros	**Resultados**
network type	RNN
sigmoid	logistica
learningrate	0.8
hidden-dim	c(15)
numepochs	10
batch size	1

Tabela 9.5: Hiperparâmetros do modelo da rede neuronal recorrente com o algoritmo RNN, utilizando a série produto interno bruto de portugal

Como pode-se verificar o erro do modelo vai diminuindo à medida que se vai aumentando o número de época de treinamento,e essas descidas são constantes com o aumento do número de épocas de treinamento, como mostra o gráfico 9.15.

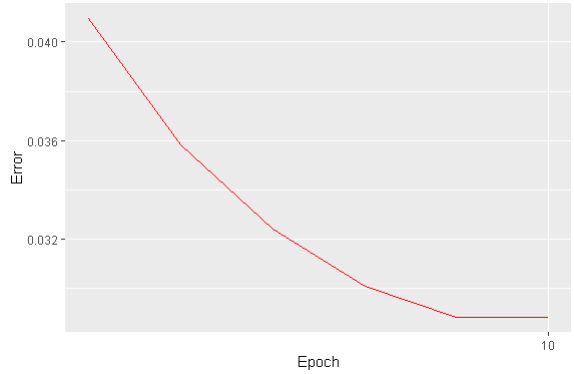

Figura 9.15: Evolução do erro por época com o uso da rede neuronal recorrente, algoritmo (RNN) utilizando a série produto interno bruto de portugal (PIB)

Quanto ao ajustamento verificamos que a rede vem aprendendo e a melhorar o ajustamento a medida que o erro vai diminuindo, mas apesar de ter uma pequena percepção do padrão dos dados na parte final da série, não teve um ajustamento bom. O modelo das redes neuronais recorrentes utilizando o algoritmo RNN obteve uma acurácia de ajustamento MAPE = 1.9849, ou seja, através do aprendizado de máquina com o algoritmo RNN, o modelo criado tem um acerto no ajustamento de 98,02%. A figura 9.16 nos ilustra essa evolução no aprendizado supervisionado utilizando o algoritmo RNN.

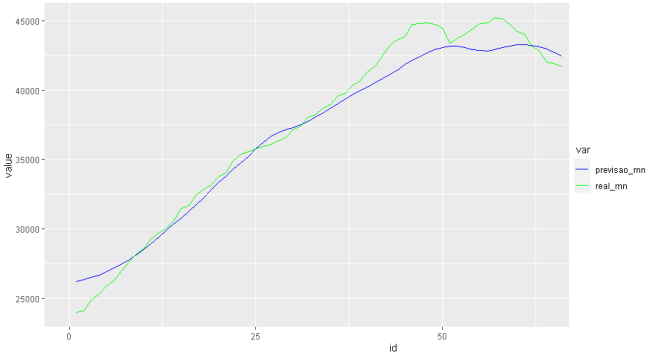

Figura 9.16: Ajustamento da série produto interno bruto de portugal utilizando as redes neuronais recorrentes com o algoritmo RNN

Utilizando o algoritmo GRU, com a função de ativação tangente hiperbólica, com uma taxa de aprendizado de 0.8, um treinamento com 60 retroprogação e um unico lote, resumidos na tabela 9.6 verificou-se também que assim como o algoritmo RNN, o erro vai diminuindo a medida que se vai treinando, contudo com mais oscilações durante todo o treinamento, como mostra a figura 9.17.

Hiperparâmetros	**Resultados**
network type	GRU
sigmoid	tanh
learningrate	0.8
hidden-dim	c(16)
numepochs	60
update-rule	sgd
batch size	1

Tabela 9.6: Hiperparâmetros do modelo da rede GRU da série produto interno bruto de portugal

Como pode-se verificar o erro do modelo vai diminuindo a medida que se vai aumentando o número de treinamento por época, sendo o erro médio por época de 0.09.

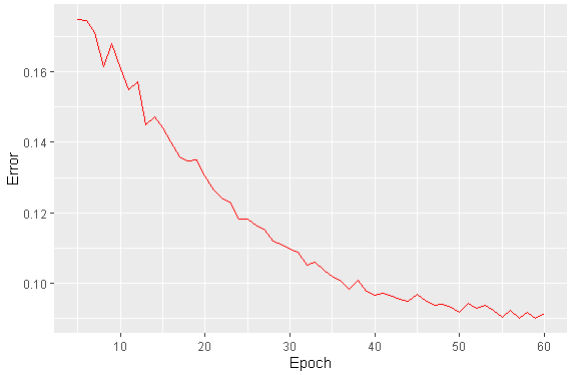

Figura 9.17: Evolução do erro por época utilizando o algoritmo GRU da série produto interno bruto de portugal

Quanto ao ajustamento o modelo das redes neuronais recorrentes utilizando o algoritmo GRU obteve uma *performance* de ajustamento, MAPE = 6.06%, ou seja, através do aprendizado de máquina com o algoritmo GRU, o modelo criado tem um acerto no ajustamento de 93.94%. A figura 9.18 nos ilustra essa evoluição no ajustamento utilizando um aprendizado supervisionado. Verifica-se que tem um fraco ajustamento na parte inicial, mas na parte final da série consegue imitar o padrão da série mesmo que ligeiramente.

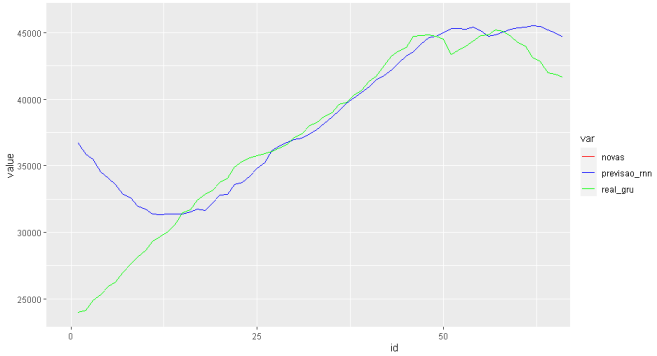

Figura 9.18: Ajustamento utilizando o algoritmo GRU da série produto interno bruto de Portugal

Para obter os resultados do ajustamento utilizando as redes neuronais artificiais e o algoritmo LSTM, utilizou-se para o treinamento os hyperparametros apresentados na tabela 9.7, cujo a função de activação foi logística, com uma taxa de aprendizagem de 0.8, para um treinamento com 100 retropropagações, utilizando um único lote.

Hyperparametros	**Resultados**
network type	LSTM
sigmoid	logistic
learningrate	0.8
hidden-dim	c(15)
numepochs	100
batch size	1

Tabela 9.7: Hiperparâmetros do modelo utilizando o algoritmo LSTM da série produto interno bruto de Portugal

O erro do modelo vai diminuindo com o aumentando do número de treinamento por época, onde se obteve um erro médio por época igual a 0.08, representado na figura 9.19 abaixo.

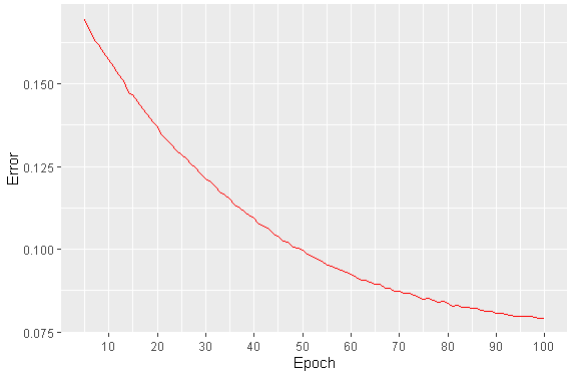

Figura 9.19: Evolução do erro por época utilizando o algoritmo LSTM da série produto interno bruto de Portugal

Quanto ao ajustamento a rede vem aprendendo e a melhorar o ajustamento a medida que o erro vai diminuindo. Na parte final da série o ajustamento diminui embora compreendendo de forma suave o padrão dos dados. O modelo das redes neuronais recorrentes utilizando o algoritmo lstm obteve uma acurácia de ajustamento MAPE = 5.39, ou seja, através do aprendizado de máquina com o algoritmo LSTM, o modelo criado tem um acerto no ajustamento de 94,62%. A figura 9.20 nos ilustra essa evolução no aprendizado supervisionado.

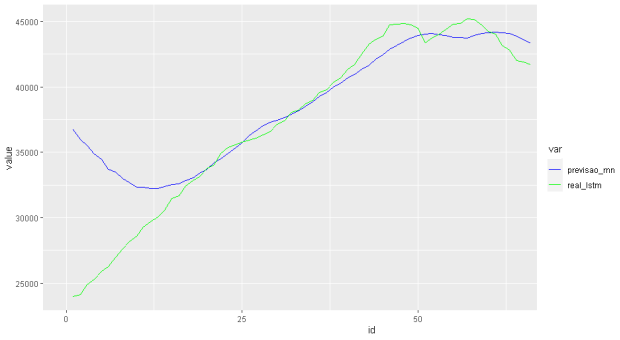

Figura 9.20: Ajustamento utilizando o algoritmo LSTM da série produto interno bruto de portugal

Resumindo os resultados utilizando os diferentes métodos de ajustamento utilizado constata-se um melhor desempenho do modelo linear, seguido da seguida RNN, onde apresentaram um erro de ajustamento inferior a 2%. Enquanto que as redes GRU e LSTM apresentaram um erro de ajustamento superior a 6.06% e 5.53% respectivamente.

	ME	RMSE	MAE	MPE	MAPE
ARIMA	39.79	291.48	222.62	0.08	0.60
RNN	-293.882	931.72	730.08	0.94	1.98
GRU	1902.31	3747.70	2236.22	5.26	6.06
LSTM	-1223.70	3537.07	2009.28	3.60	5.53

Tabela 9.8: Resumo dos resultados dos diferentes modelos utilizando diferentes algoritmos e o método estatístico da série produto interno bruto de Portugal

9.1.3 Lago Huron do ano de 1860 até ao ano 1937

Os dados da série Lago Huron (localizado no estado do Michigan, EUA) dizem respeito à elevação média da superfície da água em julho (medidos em pés), iniciando no ano de 1860 até ao ano de 1937. Graficamente verifica que a série possui uma tendência decrescente.

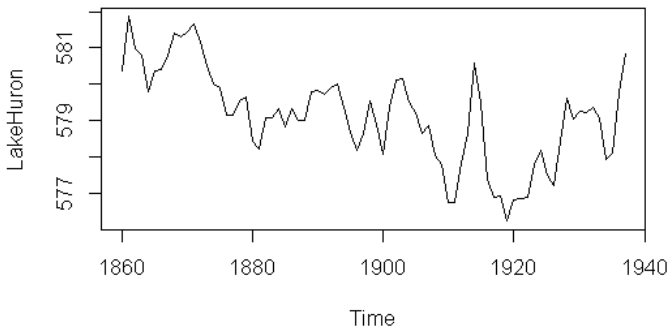

Figura 9.21: Série lago Huron entre 1860-1937

Verifica-se que a série do Lago Huron possui tendência decrescente - teste de tendência de Mann-Kendall ($P - value < 0.05$) e teste de Cox-Stuart ($P - value < 0.05$).

Dado que a série não é estacionária estimou-se o parâmetro da transformação de Box-Cox, tendo-se obtido $\tilde{\lambda} = 1$, ou seja, não foi necessário transformar a série. Para estacionarizar a série foram feitas somente duas diferenciações simples, cujas FAC e FACP da série Lake Huron se apresentam de seguida da figura 9.22.

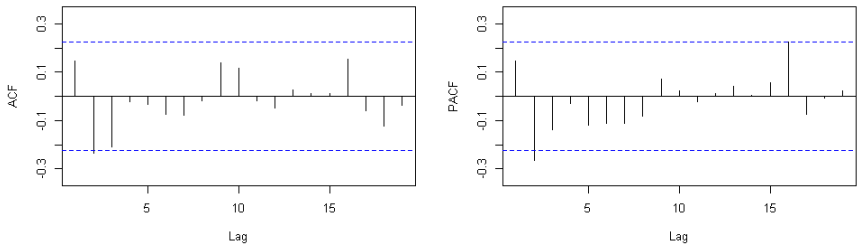

Figura 9.22: Função de auto-correlação e autocorrelação parcial da série transformada do Lago Huron

Quanto ao modelo encontrado foi um ARIMA(2,1,0). Os resíduos do modelo podem considerar-se ruído branco, representados pelas FAC e FACP, apresentado na figura 9.23 seguindo uma distribuição normal onde temos o teste de normalidade de Kolmogorov-Smirnov($P - value = 0.99$), com a média zero, com um ($P - value = 0.94$) para o teste t.

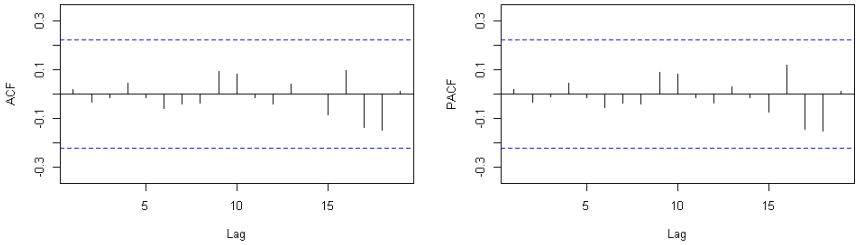

Figura 9.23: Residuos da série Lago Huron utilizando o modelo auto-regressivo integrado de médias móveis

Quanto ao ajustamento, obteve-se um MAPE $= 0.09$, ou seja, o modelo teve uma performance de ajustamento de 99.9% demonstrado na figura 9.24.

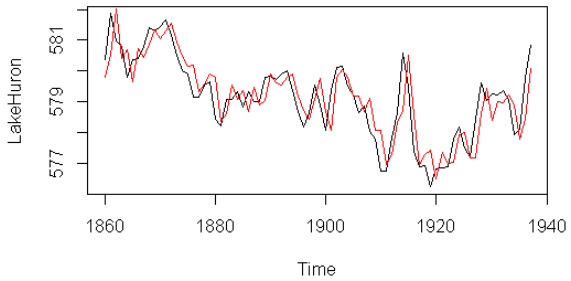

Figura 9.24: Ajustamento da série Lago Huron utilizando o modelo auto-regressivo integrado de médias móveis

Para obter os resultados do ajustamento utilizando as redes neuronais artificiais e o algoritmo RNN, utilizou-se para o treinamento os hiperparâmetros apresentados na tabela 9.9, cuja função de activação foi logística,

com uma taxa de aprendizagem de 0.5, para um treinamento com 50 retropropagação, utilizando um único lote, onde se obteve um erro médio por época igual a 0.107, representado na figura 9.25 abaixo.

Hyperparâmetros	**Resultados**
network type	RNN
sigmoid	logistica
learningrate	0.5
hidden-dim	c(10)
numepochs	50
batch size	1

Tabela 9.9: Hiperparâmetros do modelo da rede neuronal recorrente utilizando o algoritmo RNN para a série Lago Huron

Como pode-se verificar o erro do modelo vai diminuindo a medida que se vai aumentando o número de treinamento por época, mas observa-se algumas oscilações do erro a medida que se vai diminuindo.

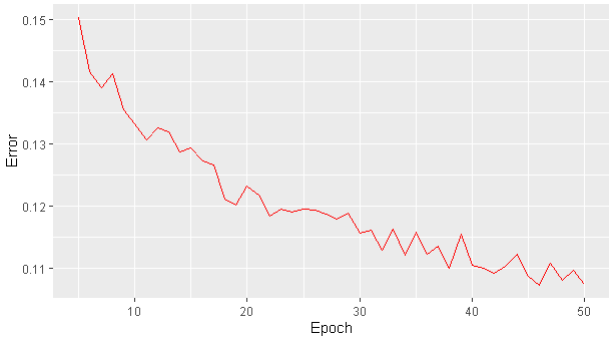

Figura 9.25: Evolução do erro por época utilizando o algoritmo das redes neuronais recorrentes RNN para a série Lago Huron

Quanto ao ajustamento verificamos que a rede vem aprendendo e a melhorar o ajustamento a medida que o erro vai diminuindo. O modelo das redes neuronais recorrentes utilizando o algoritmo RNN obteve uma acurácia de ajustamento MAPE = 0.10, ou seja, através do aprendizado de máquina com o algoritmo RNN, o modelo criado teve um acerto no ajustamento de 99,9%. A figura 9.26 nos ilustra essa evolução no ajustamento da rede utilizando o aprendizado supervisionado. Pode-se ver que mesmo sendo uma série com muitas oscilações o modelo consegue acompanhar os dados.

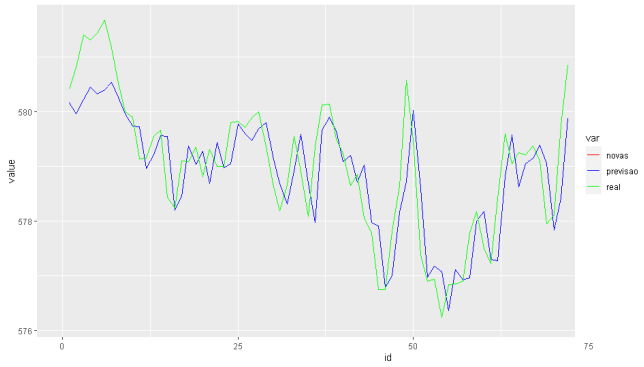

Figura 9.26: Ajustamento utilizando o algoritmo das redes neuronais recorrentes da série Lago Huron

Através do algoritmo GRU, utilizando a função de ativação logística, com uma taxa de aprendizado de 0.6, com um treinamento com 80 retroprogações e um único lote, resumidos na tabela 9.10 verificou-se também que assim como o algoritmo RNN, o erro vai diminuindo a medida que se vai treinando, mas também com muitas oscilações durante todo o treinamento, como mostra a figura 9.27.

Hiperparâmetros	**Resultados**
network type	GRU
sigmoid	logistic
learningrate	0.6
hidden-dim	c(10)
numepochs	80
update-rule	sgd
batch size	1

Tabela 9.10: Hiperparâmetros do modelo da rede GRU da série Lago Huron

Como pode-se verificar o erro do modelo vai diminuindo a medida que se vai aumentando o número de treinamento por época, sendo o erro médio por época de 0.11, mas observam-se pequenas oscilações do erro a medida que se vai diminuindo.

Figura 9.27: Evolução do erro por época utilizando o algoritmo GRU da série LakeHuron

Quanto ao ajustamento verificamos que a rede vem aprendendo e a melhorar o ajustamento a medida que o erro vai diminuindo, efetuando posteriormente uma previsão. O modelo das redes neuronais recorrentes utilizando o algoritmo GRU obteve uma acurácia de ajustamento MAPE =0.107, ou seja, através do aprendizado de máquina com o algoritmo GRU, o modelo criado tem um acerto no ajustamento de 99.9%. A figura 9.28 nos ilustra essa evolução com um aprendizado supervisionado.

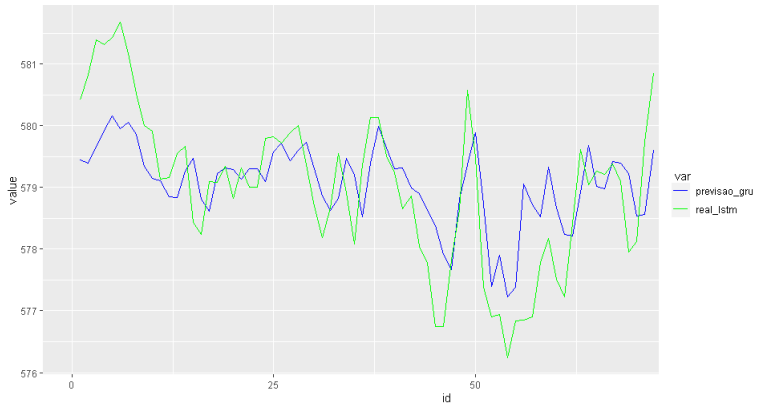

Figura 9.28: Ajustamento utilizando o algoritmo GRU da série Lago Huron

Para obter os resultados do ajustamento utilizando as redes neuronais artificiais e o algoritmo LSTM, utilizou-se para o treinamento os hiperparâmetros apresentados na tabela 9.11, cuja função de ativação foi logística, com uma taxa de aprendizagem de 0.8, para um treinamento com 200 retropropagações, utilizando um único lote.

Hiperparâmetros	**Resultados**
network type	LSTM
sigmoid	logistic
learningrate	0.8
hidden-dim	c(10)
numepochs	200
batch size	1

Tabela 9.11: Hiperparâmetros do modelo da rede utilizando o algoritmo LSTM para a série Lake Huron

Obteve um erro médio por época igual a 0.108, representado na figura 9.29 abaixo. Como pode-se verificar o erro do modelo vai diminuindo a medida que se vai aumentando o número de treinamento por época, mas observam-se pequenas oscilações do erro durante o treinamento.

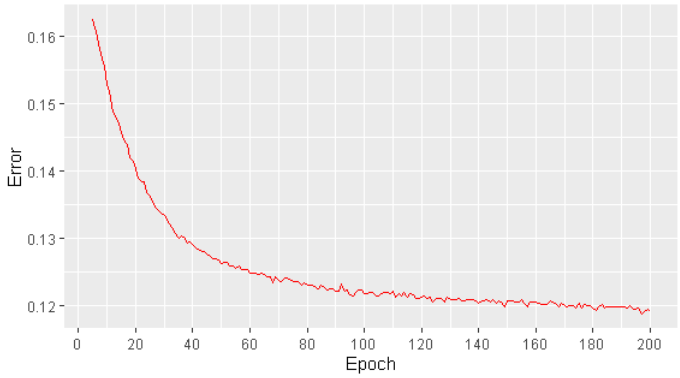

Figura 9.29: Evolução do erro por época utilizando o algoritmo LSTM com a série Lake Huron

Quanto ao ajustamento verificamos que a rede vem aprendendo e a melhorar o ajustamento a medida que o erro vai diminuindo, efetuando posteriormente uma previsão. O modelo das redes neuronais recorrentes utilizando o algoritmo lstm obteve uma acurácia de ajustamento MAPE = 0.12 ou seja, através do aprendizado de máquina com o algoritmo LSTM, o modelo criado tem um acerto no ajustamento de 99,88%. A figura 9.30 nos ilustra essa evolução no aprendizado supervisionado.

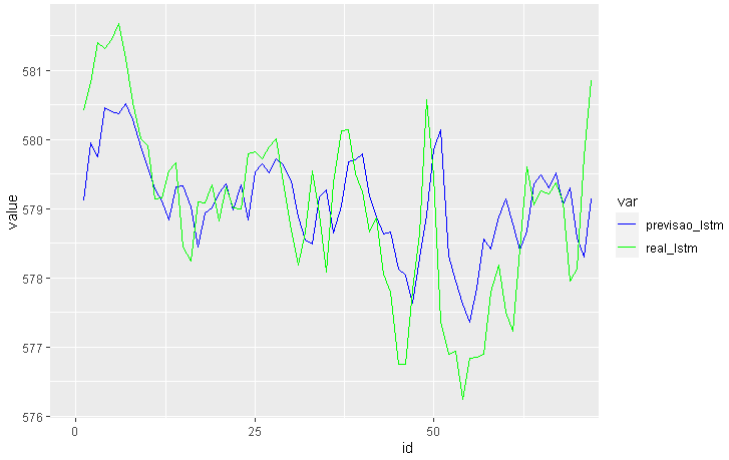

Figura 9.30: Ajustamento utilizando o algoritmo das redes LSTM da série Lago Huron

O resume na tabela 9.24 apresenta os diferentes métodos de ajustamento utilizado, assim como para os resultados da série consumo público, se constata que o modelo linear (ARIMA) obteve a melhor performance de ajustamento. Verifica-se também que no caso de séries que apresentam muitas oscilações a diferença de performance de ajustamento entre os modelos lineares e as redes neuronais para qualquer que seja o algoritmo utilizado é residual, demostrando que as redes conseguem aprender muito bem as oscilações dos dados.

	ME	RMSE	MAE	MPE	MAPE
ARIMA	0.01	0.67	0.54	0.00	0.09
RNN	-0.37	0.74	0.61	-0.06	0.11
GRU	0.22	0.70	0.57	0.04	0.10
LSTM	0.05	0.81	0.68	0.01	0.12

Tabela 9.12: Resumo dos resultados dos diferentes modelos da série Lago Huron

9.1.4 Preço de uma dúzia de ovos nos Estados Unidos entre 1900 e 1967

Os dados do preço de uma dúzia de ovos nos Estados Unidos entre 1900 e 1967, com observações anuais. Graficamente verifica que a série possui uma tendência decrescente com $(P - value = 4.5e - 10)$ e $(P - value = 1)$ obtidos através dos testes de Mann-Kendall e Cox-Stuart, respectivamente.

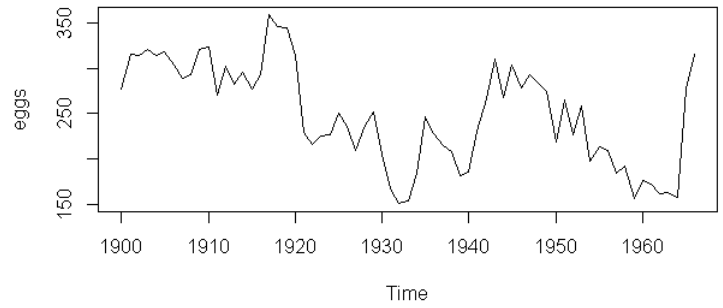

Figura 9.31: Série do preço de uma dúzia de ovos nos Estados Unidos entre 1900 e 1967

Dado a não estacionariedade da série, efetuou-se uma transformação de Box-Cox $(\tilde{\lambda} = 0.629)$ e uma diferenciação simples, tendo-se obtido as seguintes funções de autocorrelação (FAC) e autocorrelação parcial (FACP)

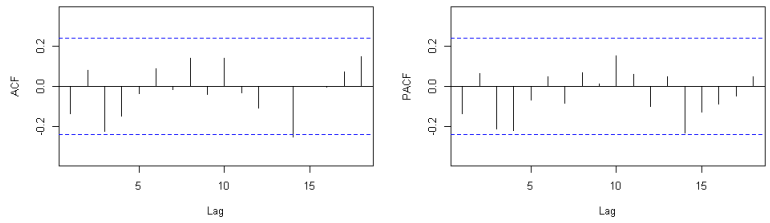

Figura 9.32: Função de auto-correlação e auto-correlação parcial da série transformada do preço de uma dúzia de ovos nos Estados Unidos

Os resíduos do modelo podem considerar-se ruído branco, representados pelas FAC e FACP, apresentado na figura 9.33 seguindo uma distribuição normal onde temos o teste de normalidade de Kolmogorov-Smirnov$(P - value = 0.96)$, com a média zero, com um $(P - value = 0.53)$ para o teste t.

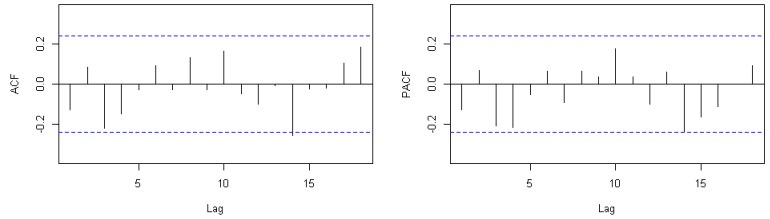

Figura 9.33: Residuo do modelo auto-regressivo integrado de médias móveis (Arima) da série preço de uma dúzia de ovos nos Estados Unidos

Quanto ao ajustamento o modelo encontrado foi um ARIMA(0,1,0) com um MAPE = 9.60. Ou seja, o modelo teve uma performance de ajustamento de 90.4%, demonstrado na figura 9.34.

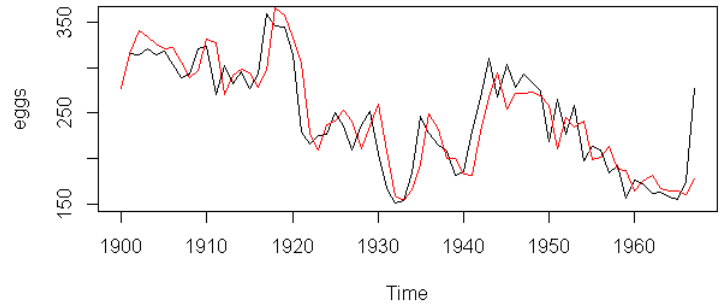

Figura 9.34: Ajustamento do modelo arima da série preço de uma dúzia de ovos nos Estados Unidos

Para obter os resultados do ajustamento utilizando as redes neuronais artificiais e o algoritmo RNN, utilizou-se para o treinamento os hiperparâmetros apresentados na tabela 9.13, cuja função de ativação foi a logística, com uma taxa de aprendizagem de 0.8, para um treinamento com 50 retropropagações, utilizando um único lote, onde se obteve um erro médio por época = 0.08, representado na figura 9.35 abaixo.

Hiperparâmetros	**Resultados**
network type	RNN
sigmoid	logistica
learningrate	0.8
hidden-dim	c(10)
numepochs	50
batch size	1

Tabela 9.13: Hiperparâmetros do modelo da utilizando o algoritmo RNN para série preço de uma duzia de ovos nos Estados Unidos

Como pode-se verificar o erro do modelo vai diminuindo a medida que se vai aumentando o número de treinamento por época, mas observam-se algumas oscilações do erro.

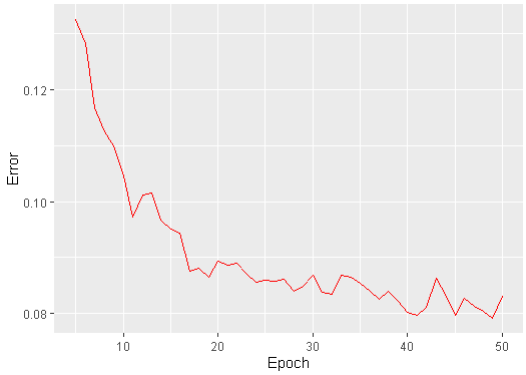

Figura 9.35: Evolução do erro por época utilizando o algoritmo RNN para a série preço de uma dúzia de ovos nos Estados Unidos

Quanto ao ajustamento verificamos que a rede vem aprendendo e a melhorar o ajustamento a medida que o erro vai diminuindo. O modelo das redes neuronais recorrentes utilizando o algoritmo RNN obteve uma acurácia de ajustamento MAPE = 10.36, ou seja, através do aprendizado de máquina com o algoritmo RNN, o modelo criado tem um acerto no ajustamento de 89,63%.

A figura 9.36 nos ilustra essa evolução no ajustamento utilizando o aprendizado supervisionado.

Figura 9.36: Ajustamento da série preço de uma dúzia de ovos nos Estados Unidos utilizando o algoritmo das redes neuronais recorrentes

Utilizando o algoritmo GRU, e a função de ativação a logística, com uma taxa de aprendizado de 0.8, com um treinamento com 25 retropropagações e um único lote, com 17 dimensões ocultas, resumidos na tabela 9.14 verificou-se também que assim como o algoritmo RNN, o erro vai diminuindo a medida que se vai treinando, mas também com oscilações durante o treinamento, como mostra a figura 9.37.

Hiperparâmetros	Resultados
network type	GRU
sigmoid	logistica
learningrate	0.8
numepochs	25
hidden-dim	(17)
update-rule	sgd
batch size	1

Tabela 9.14: Hiperparâmetros do modelo da rede GRU da série preço de uma dúzia de ovos nos Estados Unidos

Como pode-se verificar o erro do modelo vai diminuindo a medida que se vai aumentando o número de treinamento por época, sendo o erro médio por época de 0.09, mas observam-se pequenas oscilações do erro a medida que se vai diminuindo, e de se observar o ideal e que aconteça descidas constantes sem oscilações dos erros no treinamento.

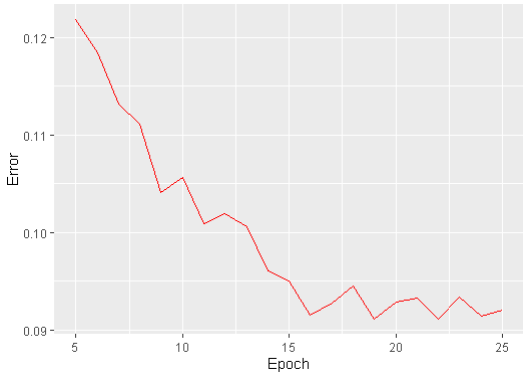

Figura 9.37: Evolução do erro por época utilizando o algoritmo GRU da série preco de uma dúzia de ovos nos Estados Unidos

Quanto ao ajustamento verificamos que a rede esta a melhorar o ajustamento, sendo que inicialmente não tinha um bom ajustamento, embora seguindo o padrão dos dados. O modelo das redes neuronais recorrentes utilizando o algoritmo GRU obteve uma acurácia de ajustamento MAPE =11.68, ou seja, através do aprendizado de máquina com o algoritmo GRU, o modelo criado tem um acerto no ajustamento de 88.32%.

A figura 9.38 nos ilustra esse ajustamento.

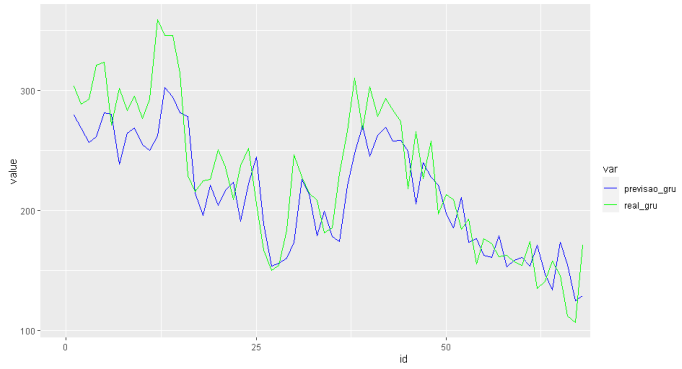

Figura 9.38: Ajustamento da série preço de uma dúzia de ovos nos Estados Unidos utilizando o algoritmo GRU

Quanto ao algoritmo LSTM, utilizou-se para o treinamento os hiperparâmetros apresentados na tabela 9.15, onde a função de ativação foi logística, com uma taxa de aprendizagem de 0.8, para um treinamento com 180 retropropagações, utilizando um único lote, onde se obteve um erro médio por época = 0.09, representado na figura 9.39 abaixo

Hiperparâmetros	**Resultados**
network type	LSTM
sigmoid	logistic
learningrate	0.8
hidden-dim	c(17)
numepochs	180
batch size	1

Tabela 9.15: Hiperparâmetros do modelo da rede LSTM da série preço de uma dúzia de ovos nos Estados Unidos

Como pode-se verificar o erro do modelo vai diminuindo a medida que se vai aumentando o número de treinamento por época.

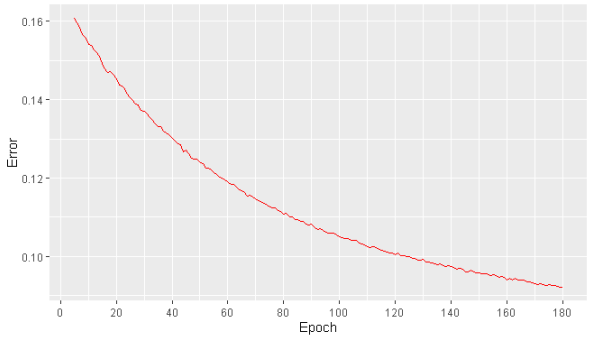

Figura 9.39: Evolução do erro por época do algoritmo LSTM da série preço de uma dúzia de ovos nos Estados Unidos

Quanto ao ajustamento verificamos que a rede vem aprendendo e a melhorar o ajustamento a medida que o erro vai diminuindo. O modelo das redes neuronais recorrentes utilizando o algoritmo RNN obteve uma acurácia de ajustamento MAPE $= 12.69$, ou seja, através do aprendizado de máquina com o algoritmo LSTM, o modelo criado tem um acerto no ajustamento de 87,31%. A figura 9.40 nos ilustra o ajustamento da série.

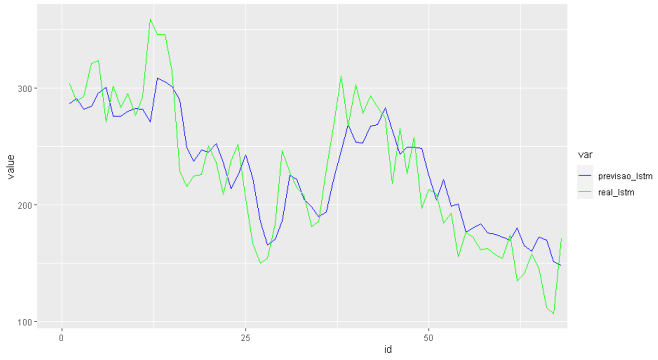

Figura 9.40: Ajustamento da série preço de uma dúzia de ovos nos Estados Unidos utilizando o algoritmo LSTM

Quanto aos diferentes resultados se constata que o modelo linear (ARIMA) estimado foi um ruído branco, não obstante teve uma boa performance de ajustamento. As redes praticamente em todos os algoritmos tiveram resultados próximos entre si e com o modelo linear. Isso quer dizer que quando se ajusta um modelo ARIMA com p=q=0, estes modelos revelaram-se muito bons no ajustamento mas seriam muito pouco realistas a prever porque as previsões seriam constantes. Enquanto que com as redes as previsões seriam muito mais realistas porque porque a estrutura das redes é de compreender o padrão da série.

	ME	RMSE	MAE	MPE	MAPE
ARIMA	0.53	29.52	22.29	-0.54	9.6
RNN	7.64	30.87	24.47	3.35	10.87
GRU	-1.16	31.97	26.00	0.1	11.68
LSTM	0.92	33.27	27.19	1.48	12.03

Tabela 9.16: Resumo dos resultados dos diferentes modelos da série preço de uma dúzia de ovos nos Estados Unidos

9.1.5 Temperatura do corpo de castor do primeiro trimestre de 1990, até ao quarto trimestre de 2000

Os dados da série castor2, representam a temperatura do corpo de 4 castores fêmeas canadianas no centro-norte de Wisconsin. A temperatura corporal foi medida por telemetria a cada 10 minutos. Os dados começam no primeiro trimestre de 1990, até ao quarto trimestre de 2000. Gráficamente verifica-se que a série possui uma tendência crescente, como mostra a figura 9.41.

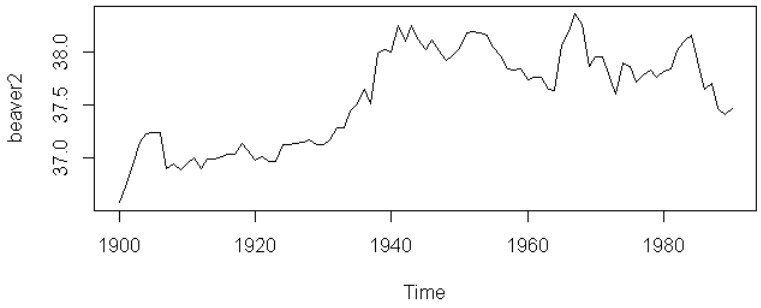

Figura 9.41: Série castor 2 entre o primeiro trimestre de 1990 e quarto trimestre de 2000

Analisando pelo modelo linear, verificamos que a série castor2 (que possui tendência crescente), onde o teste de tendência de Mann-Kendall ($P-value = 6.093e-10$), e o teste de Cox-Stuart ($P-value = 1$). Efetuou-se uma transformação de Box-Cox, por a série não ser estacionária, onde $\hat{\lambda} = -1$, e para estacionarizar a série foi feita uma diferenciação simples.

As funções de autocorrelação e autocorrelação parcial da série castor2 são apresentadas na figura 9.42

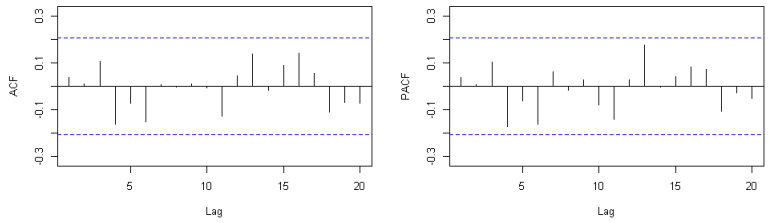

Figura 9.42: Função autocorrelação e autocorrelação parcial da série transformada do castor2

O melhor modelo encontrado foi um ARIMA(0,1,0). Os resíduos do modelo podem considerar-se ruído branco, representados pelas FAC e FACP, apresentado na figura 9.43 nao seguindo uma distribuição normal onde temos o teste de normalidade de Kolmogorov-Smirnov($P - value = 0.01$), mas com a média zero, com um ($P - value = 0.23$) para o teste t..

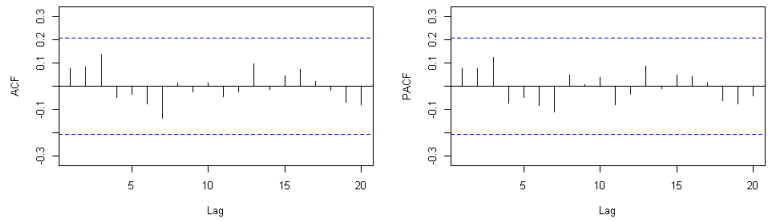

Figura 9.43: Residuos da série castor2

Quanto ao ajustamento, obteve-se um MAPE = 0.29, ou seja, o modelo teve uma performance de ajustamento de 99.7%, demonstrado na figura 9.44.

Para obter os resultados do ajustamento utilizando as redes neuronais artificiais e o algoritmo RNN, utilizou-se para o treinamento os hiperparâmetros apresentados na tabela 9.17, cuja função de ativação foi a

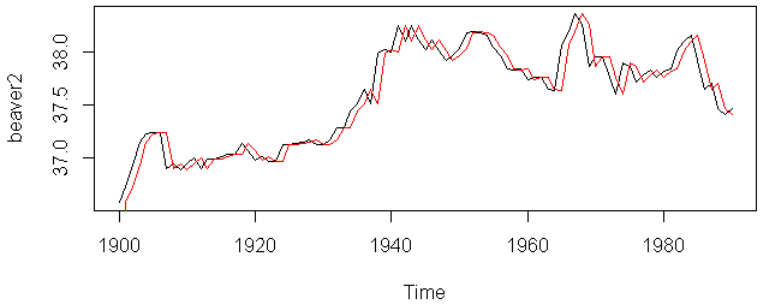

Figura 9.44: Ajustamento utilizando o modelo auto-regressivo integrado de médias moveis da série produto interno bruto de portugal

logística, com uma taxa de aprendizagem de 0.5, para um treinamento com 100 retropropagações, utilizando um unico lote, onde se obteve um erro médio por época = 0.0645, representado na figura 9.45 abaixo.

Hiperparâmetros	**Resultados**
network type	RNN
sigmoid	logistica
learningrate	0.5
numepochs	100
batch size	1

Tabela 9.17: Hiperparâmetros do modelo da rede neuronal recorrente da série castor2

Como pode-se verificar o erro do modelo vai diminuindo a medida que se vai aumentando o número de treinamento por época, com pequenas oscilações do erro, até atingir o mínimo.

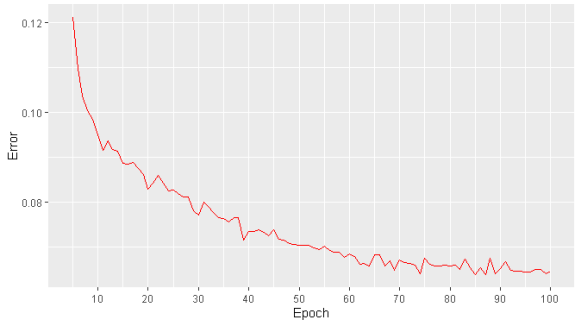

Figura 9.45: Evolução do erro por época da variavel castor2 utilizando o algoritmo das redes neuronais recorrentes

Quanto ao ajustamento verifica-se que a rede vem aprendendo e a acompanhar as várias oscilações da série, sendo essa série com diversas oscilações dificultando o ajustamento. Mas é de grande importância realçar que a rede consegue compreender o padrão da série.

Utilizando o algoritmo RNN obteve-se uma acurácia de ajustamento MAPE $= 0.25$, ou seja, através do aprendizado de máquina com o algoritmo RNN, o modelo criado tem um acerto no ajustamento de 99.75%. A figura 9.46 nos ilustra essa evolução no ajustamento utilizando o aprendizado supervisionado.

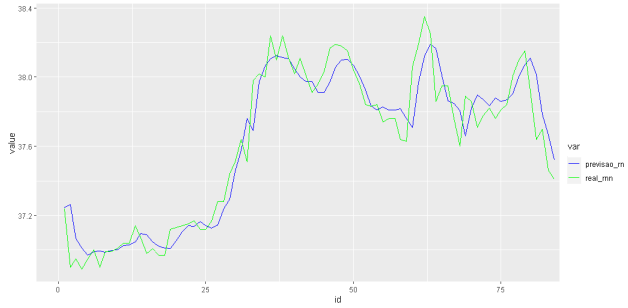

Figura 9.46: Ajustamento utilizando o algoritmo das redes neuronais recorrentes da série castor2

Utilizando um outro algoritmo GRU, e os hiperparâmetros resumidos na tabela 9.18, cuja função de ativação foi uma logística, com uma taxa de aprendizado de 0.5, um treinamento com 35 retroprogações e um único lote, verificou-se também que assim como o algoritmo RNN, o erro vai diminuindo a medida que se vai treinando, mas também com muitas oscilações durante todo o treinamento, como mostra a figura 9.47.

Hiperparâmetros	**Resultados**
network type	GRU
sigmoid	logistic
learningrate	0.5
numepochs	35
update-rule	sgd
batch size	1

Tabela 9.18: Hiperparâmetros do modelo utilizando o algoritmo da rede GRU para a série castor2

Como pode-se verificar o erro do modelo vai diminuindo a medida que se vai aumentando o número de treinamento por época, sendo o erro médio por época de 0.09, mas observam-se pequenas oscilações do erro durante o treinamento.

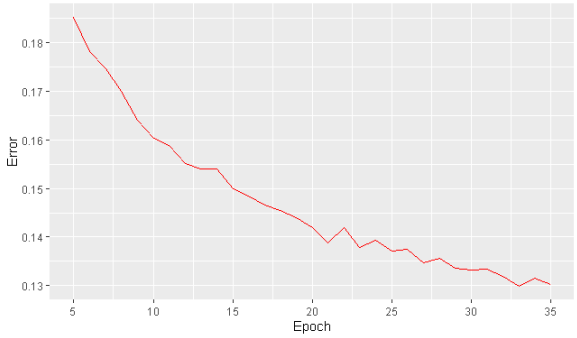

Figura 9.47: Evolução do erro por época utilizando o algoritmo das redes neuronais recorrentes da variável castor2

O modelo das redes neuronais recorrentes utilizando o algoritmo GRU obteve uma acurácia de ajustamento MAPE = 0.29, ou seja, através do aprendizado de máquina com o algoritmo RNN, o modelo criado tem um acerto no ajustamento de 99.71%, com a média dos resíduos de 0.13. Verifica-se que a rede vem aprendendo e a melhorar o ajustamento. A figura 9.48 nos ilustra essa evolução no aprendizado supervisionado.

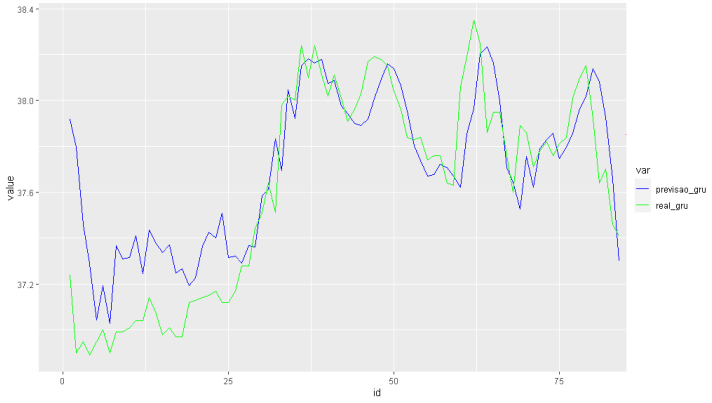

Figura 9.48: Ajustamento utilizando o algoritmo GRU para a série castor2

Quanto ao ajustamento utilizando algoritmo LSTM, utilizou-se para o treinamento os hiperparâmetros apresentados na tabela 9.19, cujo a função de ativação foi logística, com uma taxa de aprendizagem de 0.75, para um treinamento com 140 retropropagações, utilizando um único lote, onde se obteve um erro médio por época igual a 0.144, representado na figura 9.49 abaixo.

Hiperparâmetros	**Resultados**
network type	LSTM
sigmoid	logistica
learningrate	0.75
numepochs	140
batch size	1

Tabela 9.19: Hiperparâmetros do modelo utilizando a rede LSTM da série castor2

Como pode-se verificar o erro do modelo vai diminuindo a medida que se vai aumentando o número de treinamento por época, com insignificantes oscilações do erro.

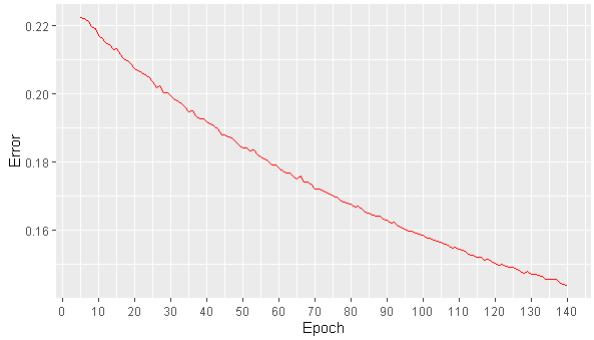

Figura 9.49: Evolução do erro por época utilizando a algoritmo LSTM da série castor2

Quanto ao ajustamento verificou-se que a rede vem melhorando o ajustamento a medida que o erro vai diminuindo, vê-se também que a série ajustada vai compreendendo o padrão da série original. O modelo das redes neuronais recorrentes utilizando o algoritmo lstm obteve uma acurácia de ajustamento MAPE = 0.48, ou seja, através do aprendizado de máquina com o algoritmo LSTM, o modelo criado tem um acerto no ajustamento de 99.5%. A figura 9.50 nos ilustra essa evolução no aprendizado supervisionado.

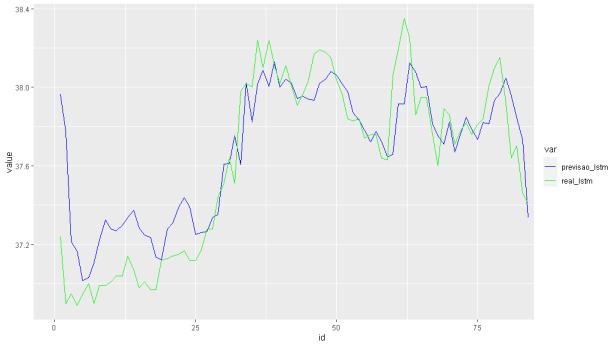

Figura 9.50: Ajustamento utilizando o algoritmo LSTM da variável castor2

Resumindo os resultados para os ajustamentos utilizados constatam-se assim como na série Lago Huron a série castor apresenta um ruído branco, resultados muito proximos quanto a acuracia de ajustamento, em que nesse caso o algoritmo RNN, obteve o melhor desempenho que o modelo linear e as outras redes utilizadas.

	ME	RMSE	MAE	MPE	MAPE
ARIMA	-0.02	0.19	0.11	-0.07	0.29
RNN	0.02	0.13	0.09	0.04	0.25
GRU	0.14	0.25	0.25	0.25	0.56
LSTM	0.07	0.23	0.18	0.19	0.48

Tabela 9.20: Resumo dos resultados dos diferentes modelos variável Castor2

9.2 Séries temporais com tendência e sazonalidade

Para a análise das séries temporais com tendência e sazonalidade, se utilizou 4 séries temporais: Venda mensal dos antigos passes urbanos L12 dos transportes de Lisboa, morte nos Estados Unidos, números de passageiros de linha aérea em Austrália e a produção trimestral de automóveis de passageiros no Reino Unido.

9.2.1 Passe - Lisboa de janeiro do ano de 1991 até fevereiro do ano de 1998

Os dados da série Passe Lisboa correspondem à venda mensal dos antigos passes urbanos L12 dos transportes de Lisboa, entre janeiro do ano de 1991 e fevereiro de 1998. Graficamente verifica-se que a série possui uma tendência decrescente com sazonalidade ($P - value < 0.05$ e $P - value = 1$ dos testes de Mann-Kendall e Cox-Stuart, respectivamente).

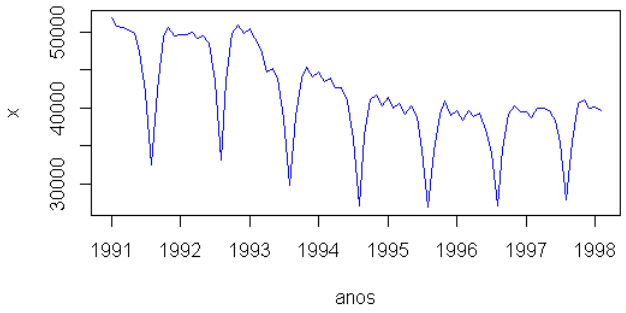

Figura 9.51: Série passe lisboa janeiro de 1991 ate fevereiro de 1998, com uma frequência mensal

Obteve-se um $\tilde{\lambda} = 1$ na transformação de Box-Cox. Não sendo necessário efectuar a transformação fez-se, contudo, uma diferenciação simples e uma diferenciação sazonal para estacionarizar a série.

As funções de autocorrelação e autocorrelação parcial da série encontrada-se apresenta na figura 9.52

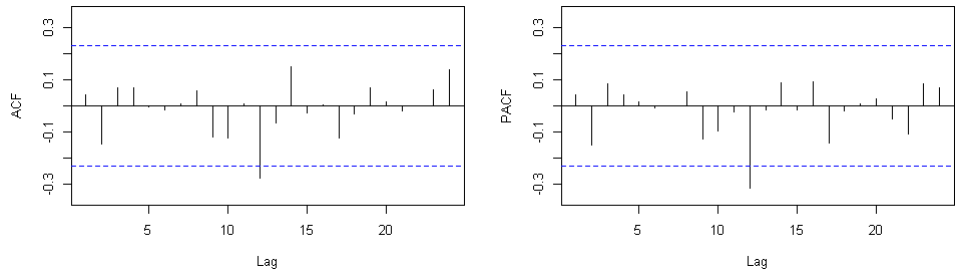

Figura 9.52: Função de autocorrelação e autocorrelação parcial da série transformada do passe Lisboa

O modelo mais adequado encontrado foi um $SARIMA(0,1,0)(1,1,0)_{[12]}$, onde todos os parâmetros do modelo foram significativos. Os resíduos do referido modelo apresentam-se como um ruído branco, apresentado na figura 9.53, nao seguindo uma distribuição normal onde temos o teste de normalidade de Kolmogorov-Smirnov($P - value = 0.027$), mas de média zero, com um ($P - value = 0.606$) para o teste t.

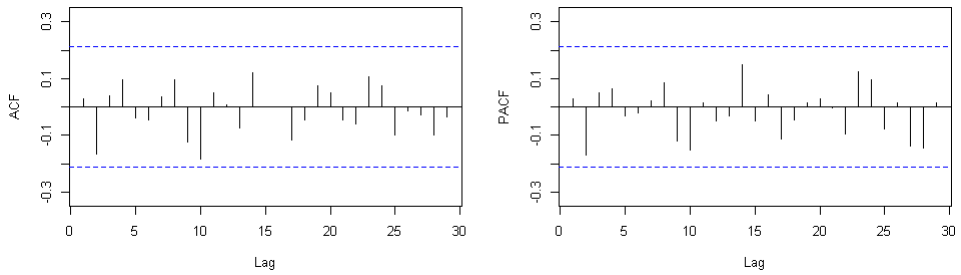

Figura 9.53: Resíduos da série passe lisboa utilizando o modelo estatístico autorregressivo integrado de médias móveis sazonais.

Quanto ao ajustamento, encontrou-se um $MAPE = 1.33$, ou seja, o modelo teve um ajustamento de 98.67%, como se apresenta na figura 9.54.

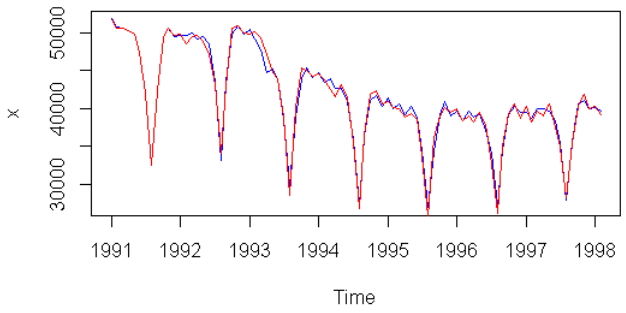

Figura 9.54: Ajustamento da série Passe Lisboa utilizando o modelo estatístico auto-regressivo de médias móveis sazonais

Para obter os resultados do ajustamento utilizando as redes neuronais artificiais e o algoritmo RNN, utilizou-se para o treinamento os hiperparâmetros apresentados na tabela 9.21, onde a função de ativação foi Gompertz, com uma taxa de aprendizagem de 0.75, para um treinamento com 100 retropropagações, utilizando um único lote, onde se obteve um erro médio por época = 0.123, representado na figura 9.55 abaixo.

Hiperparâmetros	**Resultados**
network type	RNN
sigmoid	Gompertz
learningrate	0.75
numepochs	100
batch size	1

Tabela 9.21: Hiperparâmetros do modelo da rede neuronal recorrente com o algoritmo RNN utilizando a série passe lisboa

Observa-se algumas oscilações do erro a medida que se vai aumentando a época de treinamento, com uma certa estabilização a partir da vigésima época.

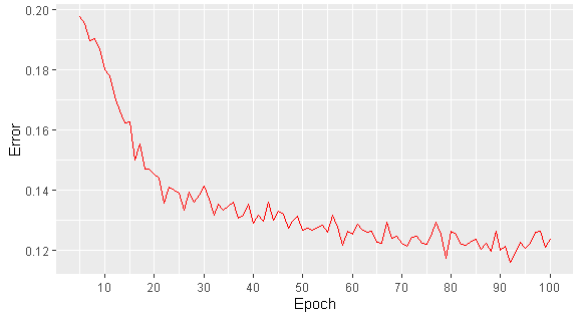

Figura 9.55: Evolução do erro por época da rede neuronal recorrente utilizando o algoritmo RNN para a série passe lisboa

Quanto ao ajustamento observa-se que o modelo vem melhorando o ajustamento, onde inicialmente não houve um bom ajustamento. O modelo das redes neuronais recorrentes utilizando o algoritmo RNN obteve uma acurácia de ajustamento com MAPE = 12.53 , ou seja, o modelo criado tem um acerto no ajustamento de 87.47%. A figura 9.56 nos ilustra essa evolução no ajustamento.

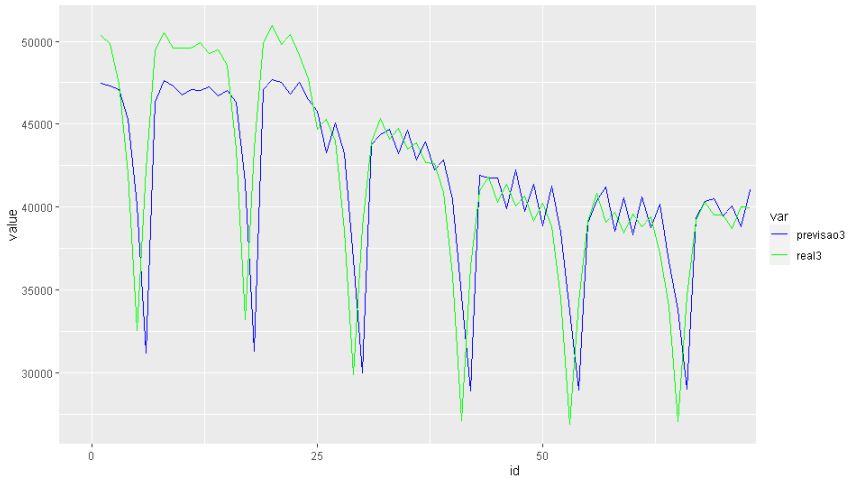

Figura 9.56: Ajustamento da série passe lisboa utilizando as redes neuronais recorrentes

Através do algoritmo GRU, e utilizando a função de ativação logística, com uma taxa de aprendizado de 0.75, com um treinamento com 150 retroprogações e um único lote como hiperparâmetros, verificou-se também que assim como o algoritmo RNN, o erro médio por época é de 0.121, que vai diminuindo a medida que se vai treinando, mas também com muitas oscilações durante todo o treinamento, como mostra a figura 9.57.

Hiperparâmetros	Resultados
network type	GRU
sigmoid	logistica
learningrate	0.75
numepochs	150
batch size	1

Tabela 9.22: Hiperparâmetros do modelo da rede GRU da série passe lisboa

Figura 9.57: Evolução do erro por época com o algoritmo gru da série passe lisboa

Quanto a acurácia do ajustamento, comparativamente com o modelo anterior tem-se um melhor resultado com um MAPE = 11.69, ou seja, houve um acerto no ajustamento de 88.31 %. O gráfico 9.58, nos mostra esse ajustamento.

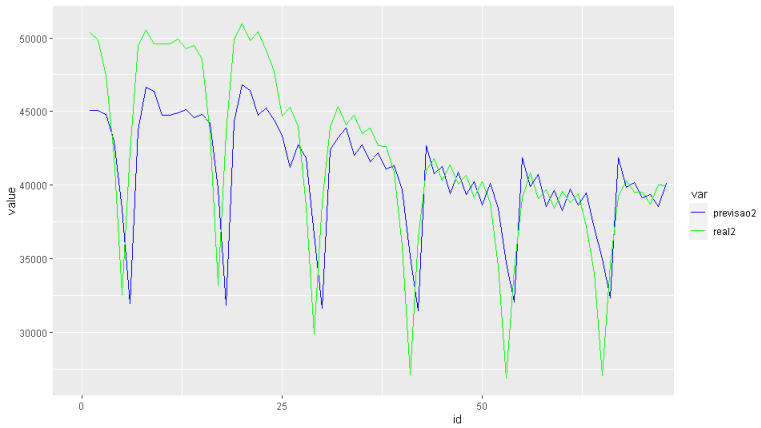

Figura 9.58: Ajustamento da série passe lisboa utilizando o algoritmo GRU

O algoritmo LSTM, comparativamente com os outros algoritmos supervisionados de aprendizado de máquina, obteve o melhor resultado, isso se observa logo no formato do erro de treinamento, em que o erro tem uma descida gradual e constante sem oscilações, ate atingir o mínimo de erro para o treinamento da série, ilustrada na figura 9.59 que e de 0.15. Quanto aos hiperparâmetros utilizados, foram a função de ativação, uma logística, com uma taxa de aprendizado por época de 0.8, para os 110 retropropagação efetuadas, e utilizou-se um único lote de treinamento, resumida na tabela 9.23.

Quando ao ajustamento, obteve-se um MAPE = 10.73, ou seja, o algoritmo teve um acerto no ajustamento de quase 90%, ilustrada na figura 9.60.

Hiperparâmetros	Resultados
network type	LSTM
sigmoid	logistica
learningrate	0.8
numepochs	110
batch size	1

Tabela 9.23: Hiperparâmetros do modelo da rede LSTM da série passe lisboa

Figura 9.59: Evolução do erro por época da série passe lisboa utilizando o algoritmo LSTM

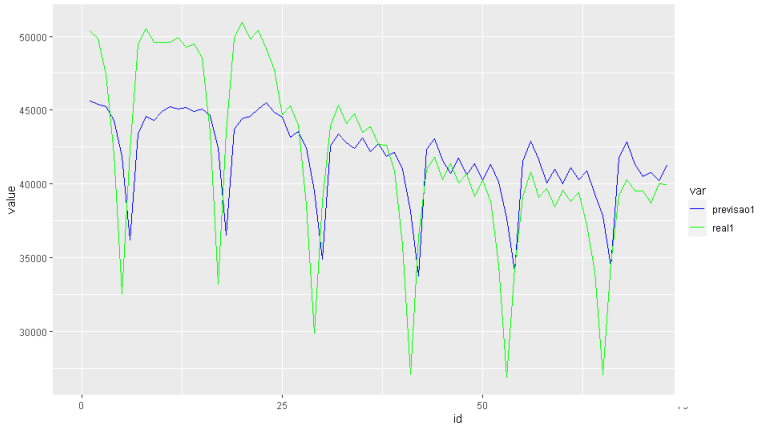

Figura 9.60: Ajustamento da série passe lisboa utilizando o algoritmo LSTM

Quanto aos resultados e resumidos na tabela 9.24 utilizando os diferentes métodos de ajustamento utilizado constata-se que o método estatístico através do modelo SARIMA $(0,1,0)(1,1,0)_{12}$ obtido claramente a melhor performance de ajustamento, enquanto que as redes praticamente em todos os algoritmos tiveram resultados próximos.

	ME	RMSE	MAE	MPE	MAPE
ARIMA	-40.75	717.14	525.69	-0.18	1.33
RNN	-646.84	6709.25	4874.62	-2.76	12.53
GRU	-854.98	6870.48	5016.74	-3.37	11.69
LSTM	-260.46	6308.28	4827.72	-0.80	10.73

Tabela 9.24: Resumo dos resultados dos diferentes modelos utilizados para o ajustamento da série venda de Passes em lisboa

9.2.2 Série da morte nos Estados Unidos de janeiro de 1973 até julho de 1981

A série da morte nos Estados Unidos, começam em janeiro de 1973 até julho de 1981, com observações mensais. Graficamente, figura 9.61, verifica-se que a série possui uma tendência crescente com um teste de tendência de Mann-Kendall ($p-value < 2.2e-16$), e essa tendência e crescente, ($p-value = 4.441e-16$) para o teste de Cox-Stuart.

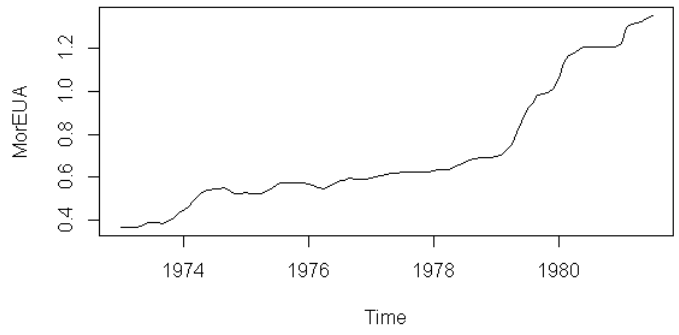

Figura 9.61: Série mensal da morte nos Estados Unidos entre janeiro de 1973 e julho de 1981

Para tornar a série estacionária efetuou-se uma transformação de Box-Cox, onde $\tilde{\lambda} = -0.161$, e uma diferenciação simples, resultando as seguintes funções de autocorrelação e autocorrelação parcial da série transformada que se apresenta na figura 9.62

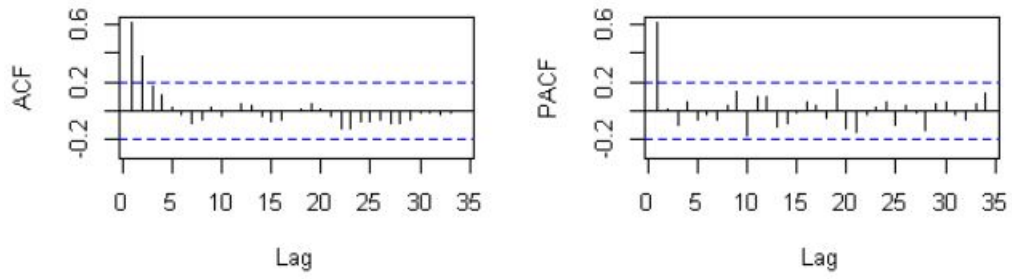

Figura 9.62: Função de autocorrelação e autocorrelação parcial da série transformada da morte nos Estados Unidos

Analisando pelo modelo linear, verifica-se que o modelo mais adequado foi um $SARIMA(1,1,0)(1,0,1)_{[12]}$, cujos resíduos do referido modelo apresentam um ruído branco, apresentado na figura 9.63. Os resíduos do referido modelo apresentam-se como um ruído branco, seguindo uma distribuição normal onde temos o teste de normalidade de Kolmogorov-Smirnov ($P-value = 0.066$), de média zero, com um ($P-value = 0.966$) para o teste t.

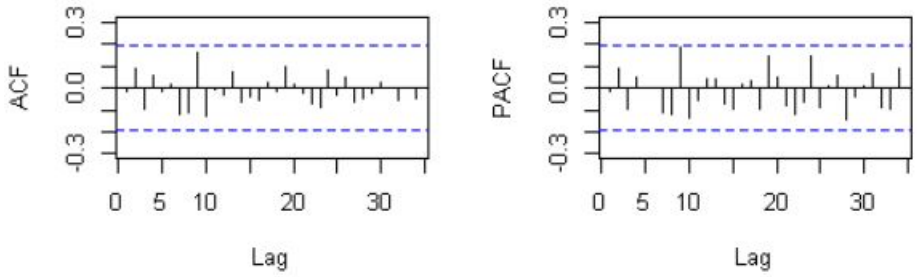

Figura 9.63: Resíduos da série morte nos Estados Unidos

Quanto ao ajustamento, obteve-se um MAPE = 1.104, ou seja, o modelo teve uma performance de ajustamento de 98.9% demonstrado na figura 9.64.

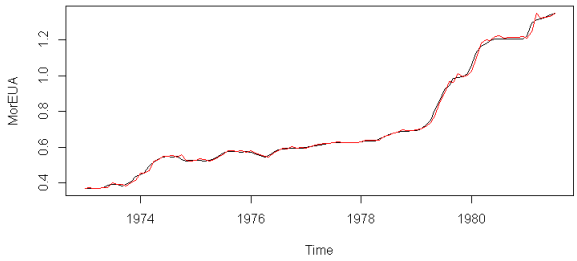

Figura 9.64: Ajustamento da série morte nos Estados Unidos

Utilizando as redes neuronais artificiais e o algoritmo RNN, utilizou-se para o treinamento os hiperparâmetros apresentados na tabela 9.25, cuja função de ativação foi a logística, com uma taxa de aprendizagem de 0.8, para um treinamento com 50 retropropagação, utilizando um único lote.

Hiperparâmetros	**Resultados**
network type	RNN
sigmoid	logistica
learningrate	0.8
hidden-dim	c(15)
numepochs	50
batch size	1

Tabela 9.25: Hiperparâmetros do modelo da rede neuronal recorrente algoritmo RNN da série morte nos Estados Unidos

Obteve o erro médio por época foi de 0.02, representado na figura 9.65 abaixo. Como pode-se verificar o erro do modelo vai diminuindo a medida que se vai aumentando o número de treinamento por época,

com pequenas oscilações do erro a medida que se vai diminuindo, e a partir da época vigésima os erros tornam-se estáveis.

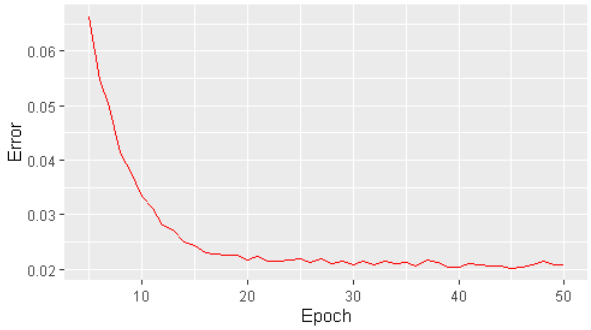

Figura 9.65: Evolução do erro por época utilizando a rede neuronal recorrente da série morte nos Estados Unidos

Quanto ao ajustamento verificamos que a rede vem aprendendo e a melhorar o ajustamento a medida que o erro vai diminuindo, até completar o ajustamento de toda a série. O modelo das redes neuronais recorrentes utilizando o algoritmo RNN obteve uma acurácia de ajustamento, MAPE = 3.63, ou seja, através do aprendizado de máquina com o algoritmo RNN, o modelo criado tem um acerto no ajustamento de 96,37%. A figura 9.66 nos ilustra esse ajustamento no aprendizado supervisionado.

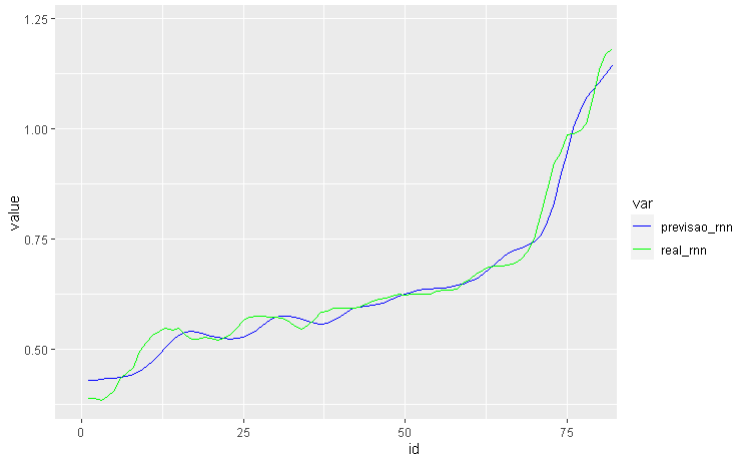

Figura 9.66: Ajustamento da série morte nos Estados Unidos utilizando o algoritmo RNN

Utilizando o algoritmo GRU, com a função de ativação logística, com uma taxa de aprendizado de 0.7, com um treinamento com 140 retropropagações e um único lote, resumidos na tabela 9.26, verificou-se também que assim como o algoritmo RNN, o erro vai diminuindo a medida que se vai treinando, mas também com pequenas oscilações durante todo o treinamento, como mostra a figura 9.67.

Hiperparâmetros	**Resultados**
network type	GRU
sigmoid	logistica
learningrate	0.7
numepochs	140
update-rule	sgd
batch size	1

Tabela 9.26: Hiperparâmetros do modelo da rede GRU morte nos Estados Unidos

Como pode-se verificar o erro do modelo vai diminuindo a medida que se vai aumentando o número de treinamento por época, sendo o erro médio por época de 0.1083.

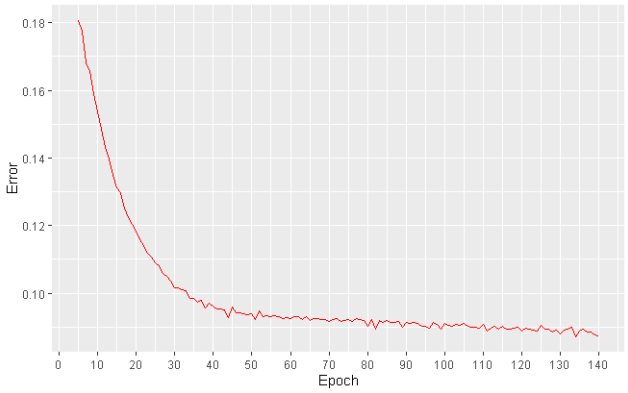

Figura 9.67: Evolução do erro por época utilizando o algoritmo GRU da série morte nos Estados Unidos

Quanto ao ajustamento verificamos que a rede vem-se ajustando a medida que se aumenta o número de treinamento e a melhorar o ajustamento a medida que o erro vai diminuindo, efetuando posteriormente uma previsão. O modelo das redes neuronais recorrentes utilizando o algoritmo GRU obteve uma acurácia de ajustamento MAPE = 12.14, ou seja, através do aprendizado de máquina com o algoritmo GRU, o modelo criado tem um acerto no ajustamento de 87,86%. A figura 9.68 nos ilustra essa evoluição no ajustamento utilizando um aprendizado supervisionado.

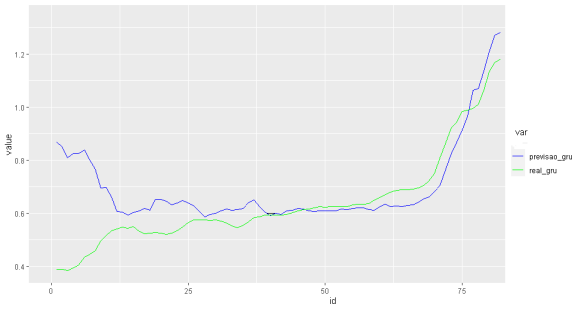

Figura 9.68: Ajustamento da série morte nos Estados Unidos utilizando o algoritmo GRU

Para obter os resultados do ajustamento utilizando as redes neuronais artificiais e o algoritmo LSTM, utilizou-se para o treinamento os hiperparâmetros apresentados na tabela 9.27, cujo a função de activação foi uma tangente, com uma taxa de aprendizagem de 0.8, para um treinamento com 380 retropropagações, utilizando um único lote, onde se obteve um erro médio por época = 0.095, representado na figura 9.69 abaixo.

Hiperparâmetros	**Resultados**
network type	LSTM
sigmoid	tanh
learningrate	0.8
numepochs	380
batch size	1

Tabela 9.27: Hiperparâmetros do modelo da rede LSTM da série morte nos Estados Unidos

Como pode-se verificar o erro do modelo vai diminuindo a medida que se vai aumentando o número de treinamento por época, mas observa-se pequenas oscilações do erro a medida que se vai diminuindo.

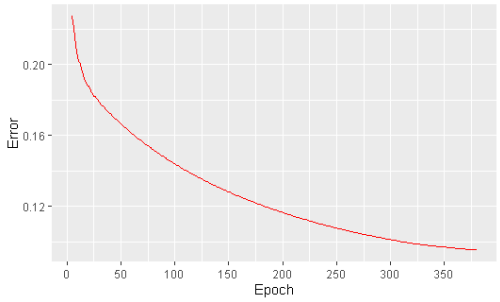

Figura 9.69: Evolução do erro por época utilizando o algoritmo LSTM da série morte nos Estados Unidos

Quanto ao ajustamento verificamos que a rede vem aprendendo e a melhorar o ajustamento a medida que o erro vai diminuindo, efetuando posteriormente uma previsão. O modelo das redes neuronais recorrentes utilizando o algoritmo lstm obteve uma acurácia de ajustamento MAPE = 12.56, ou seja, o modelo criado tem um acerto no ajustamento de 87,44%. A figura 9.70 nos ilustra essa evolução no aprendizado supervisionado.

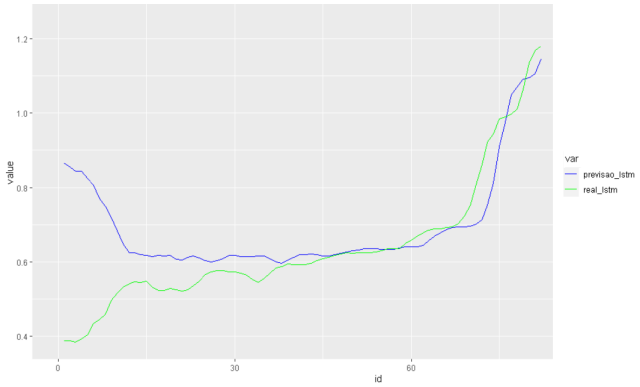

Figura 9.70: Ajustamento utilizando o algoritmo LSTM para a série morte nos Estados Unidos

Quanto aos resultados e resumido na tabela 9.24 utilizando os diferentes metodos de ajustamento utilizado constata-se que o método estatistico através do modelo SARIMA $(1,1,0)(1,0,1)_{12}$ obtido claramente a melhor performance de ajustamento, mas muito próximo está as redes com o algoritmo RNN, enquanto que as redes com o algoritmo GRU e LSTM não tiveram um bom desempenho ficando com um erro de ajuste acima dos 10%.

	ME	RMSE	MAE	MPE	MAPE
ARIMA	-1.43e-0.5	0.01202	0.0079	-0.01794	1.10358
RNN	0.00173	0.2487	0.0195	0.31709	3.202
GRU	0.04359	0.14379	0.08688	5.08813	11.63
LSTM	0.05676	0.14934	0.09207	7.63796	12.56

Tabela 9.28: Resumo dos resultados dos diferentes modelos utilizados

9.2.3 Números de passageiros de linha aérea em Austrália de janeiro de 1991 até abril de 1996

A série passagem aérea representa números de passageiros mensais de linha aérea australiana, entre janeiro de 1991 até abril de 1996.

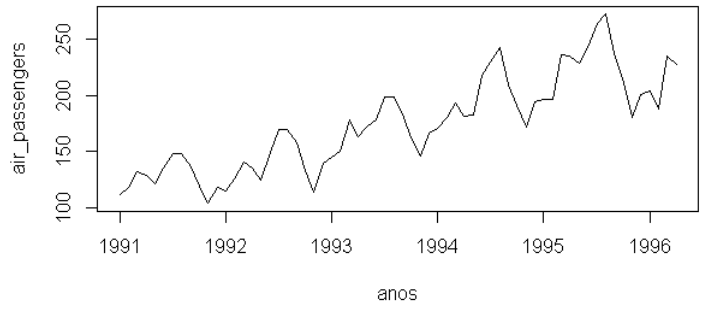

Figura 9.71: Série passagem aérea entre janeiro de 1991 e abril de 1996 com periodos sazonais
Fonte: Base de dados do *software* R

Essa série apresenta tendência e sazonalidade, com o teste de Mann-Kendall, $P - Valor = 1.776e - 15$, e teste Cox Stuart, cujo $P - Valor = 1$, ou seja, existe uma tendência crescente. Para efetuar o modelo SARIMA, efetuou-se tanto uma diferenciação simples como uma diferenciação sazonal. Para a transformação de Box-Cox foi encontrado um $\tilde{\lambda} = 0.21$.

A função de autocorrelação e auto-correlação parcial da série passagem aerea utilizando o modelo auto-regressivo integrado de médias moveis sazonais (SARIMA), apresenta-se na figura 9.72

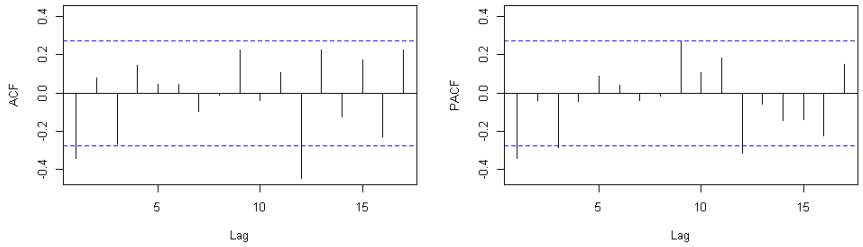

Figura 9.72: Função de autocorrelação e auto-correlação parcial da série transformada da passagem aérea

O modelo encontrado foi um SARIMA $(2,0,0)(1,1,0)_{12}$. Os resíduos do referido modelo apresentam-se como um ruído branco não seguindo uma distribuição normal onde temos o teste de normalidade de Kolmogorov-Smirnov ($P-value = 0.002$), contudo de média zero, com um ($P-value = 0.261$) para o teste t.

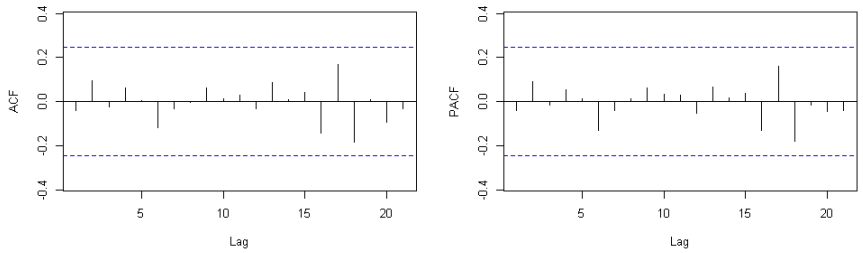

Figura 9.73: Residuos do modelo da série passagem aérea utilizando o modelo auto-regressivo integrado de médias móveis sazonais

Quanto ao ajustamento através da análise do $MAPE = 2.75$, ou seja, o modelo teve um acerto de ajustamento de 97.25%, ilustrado na figura 9.74.

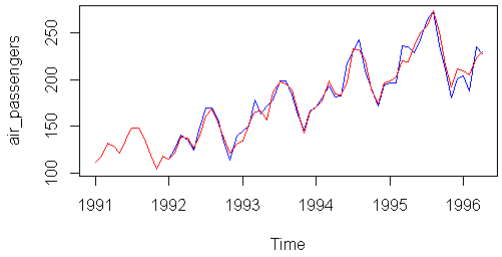

Figura 9.74: Ajustamento da série passagem aérea utilizando o modelo auto-regressivo integrado de médias móveis sazonais

Verificando as redes neuronais com o algoritmo RNN para a série passageiros da linha aérea, para o treinamento utilizou-se os seguintes hiperparâmetros, a função de ativação o logistica, com 15 dimensões escondidas, para uma taxa de aprendizado de 80%, com 90 retropropagações, para um único lote de treinamento, resumida na tabela 9.29.

Hiperparâmetros	**Resultados**
network type	RNN
sigmoid	logistica
hidden-dim	c(15)
learningrate	0.8
numepochs	90
batch size	1

Tabela 9.29: Hiperparâmetros do modelo das redes neuronais recorrentes da série passageiros da linha aérea

O erro do treinamento do modelo vai diminuindo a medida que se vão aumentando o número de épocas, ilustrado na figura 9.75 é de realçar as oscilações durante o treinamento ate atingir o minimo. Quanto ao ajustamento verifica-se que se vai melhorando mesmo que subtilmente com a época de treinamento, efetuando posteriomente uma previsão.

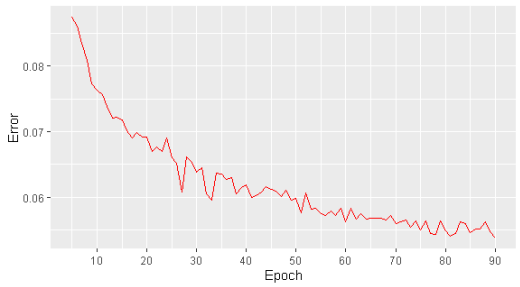

Figura 9.75: Treinamento Modelo RNN série passageiros da linha aérea

A acurácia de ajustamento foi com um $MAPE = 8.585$, ou seja, o modelo consegue com os hiperparâmetros fixados se ajustar a 91.42% aos dados, como mostra na figura 9.76.

Figura 9.76: Ajustamento da série passageiros da linha aérea utilizando o algoritmo das redes neuronais recorrentes

Utilizando alguns hiperparâmetros para o algoritmo GRU, como uma função de ativação tangente hiperbólica, com uma taxa de aprendizado de 80%, com um treinamento com 400 retroprogações e um único lote, com 14 dimensões ocultas, efetuou-se o treinamento da rede, hyperparâmetros esses resumidos na tabela 9.30.

Hiperparâmetros	**Resultados**
network type	GRU
sigmoid	tanh
hidden-dim	14
learningrate	0.8
numepochs	400
batch size	1

Tabela 9.30: Hiperparâmetros do modelo da rede GRU da série passagem aérea

Verificou-se também que tal como o algoritmo RNN, o erro vai diminuindo a medida que se vai treinando, mas também, neste caso, muitas oscilações durante todo o treinamento, até é última época de treinamento, como mostra a figura 9.77.

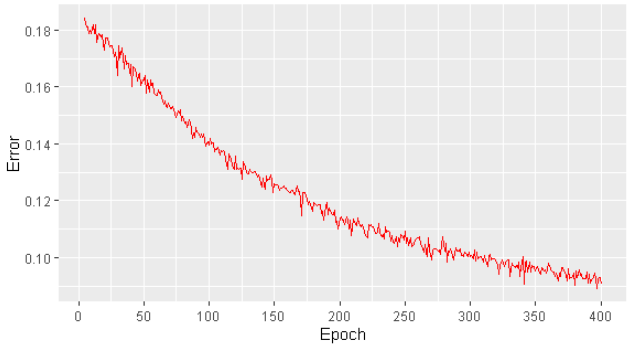

Figura 9.77: Evolução do erro por época treinamentoerroGRUAP

Quanto ao ajustamento, obteve-se um MAPE = 8.85, ou seja, o algoritmo teve um acerto no ajustamento de quase 91.15%, ilustrada na figura 9.78.

Figura 9.78: Ajustamento utilizando o algoritmo GRU da série passagem aérea

O algoritmo LSTM, comparativamente com os outros algoritmos supervisionado de aprendizado de máquina teve o pior desempenho. Quanto aos hiperparâmetros utilizados, foram a função de ativação, uma logistica, com uma taxa de aprendizado por época de 0.8, para os 250 retropropagações efetuadas, e utilizou-se um único lote de treinamento, resumida na tabela 9.31.

Hiperparâmetros	**Resultados**
network type	LSTM
sigmoid	logistic
hidden-dim	c(17)
learningrate	0.8
numepochs	250
batch size	1

Tabela 9.31: Hiperparâmetros do modelo utilizando o algoritmo LSTM da série passagem aérea

O erro de treinamento se apresenta na figura 9.79

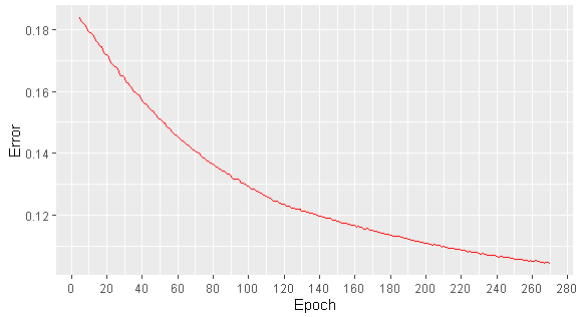

Figura 9.79: Evolução do erro por época utilizando o algoritmo LSTM da série passagem aérea

Quanto ao ajustamento, obteve um MAPE = 19.90, ou seja, o algoritmo teve um acerto no ajustamento de quase 80.1% ilustrada na figura 9.80.

Figura 9.80: Ajustamento com o algoritmo LSTM da série passagem aérea

Resumindo os resultados na tabela 9.32 verifica-se que o metodo classico SARIMA, obteve o melhor resultado com um MAPE de 2.75, e o algoritmo RNN obteve um bom resultado com um MAPE de 8.85, enquanto que os algoritmos GRU e LSTM não tiveram um bom resultado com um MAPE de 16.89 e 19.9 respectivamente.

	ME	RMSE	MAE	MPE	MAPE
ARIMA	-0.16	7.23	5.09	-0.22	2.75
RNN	-0.67	19.77	16.29	-1.83	8.85
GRU	-10.90	35.82	27.89	-9.21	16.89
LSTM	-11.63	40.55	32.30	-11.27	19.90

Tabela 9.32: Resumo dos resultados dos diferentes modelos da série passagem aérea

9.2.4 Produção trimestral de automóveis de passageiros no Reino Unido de janeiro de 1977 até janeiro de 2000

Uma outra série que apresenta tendência e sazonalidade e a produção trimestral de automóveis de passageiros no Reino Unido (em milhares de automóveis). Verifica-se que a série apresenta sazonalidade e tendência crescente, a série se inicia em janeiro do ano de 1977 ate janeiro do ano de 2000.

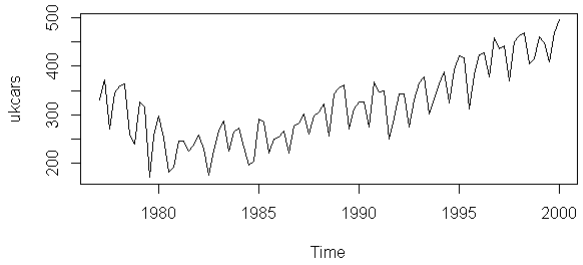

Figura 9.81: Série original da produção trimestral dos carros no Reino Unido
Fonte: Base de dados do *Software* R

Essa série é portanto uma série que apresenta tendência e sazonalidade, com o teste de Mann-Kendall ($P - Valor = 2.887e - 15$), e teste Cox-Stuart, cujo ($P - Valor = 1.551e - 07$), ou seja, existe uma tendência crescente. Para efetuar o modelo, diferenciou-se a série uma vez, tanto uma diferenciação simples como uma diferenciação sazonal. Para a transformação de box-cox foi encontrado um $\tilde{\lambda} = 0.858$.

A função de autocorrelação e autocorrelação parcial da série da produção dos carros no Reino Unidos e apresenta-se na figura 9.82

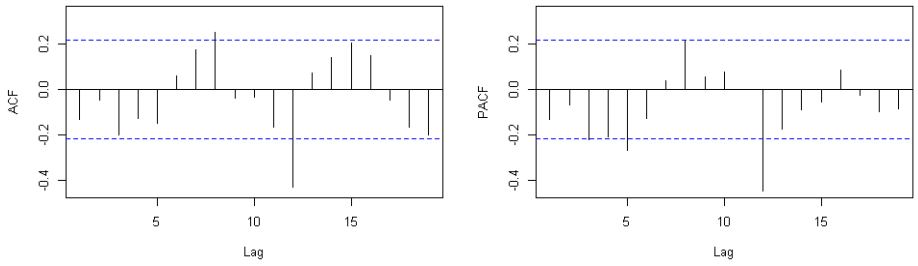

Figura 9.82: Função de auto-correlação e auto-correlação parcial da série da produção dos carros no Reino Unido

O modelo encontrado foi um SARIMA$(3,1,2)(1,1,0)_{12}$, sendo que Os resíduos do referido modelo apresentam-se como um ruído branco nao seguindo uma distribuição normal onde temos o teste de normalidade de

Kolmogorov-Smirnov ($P - value = 0.002$), contudo de média zero, com um ($P - value = 0.80$) para o teste t, apresentado na figura 9.83.

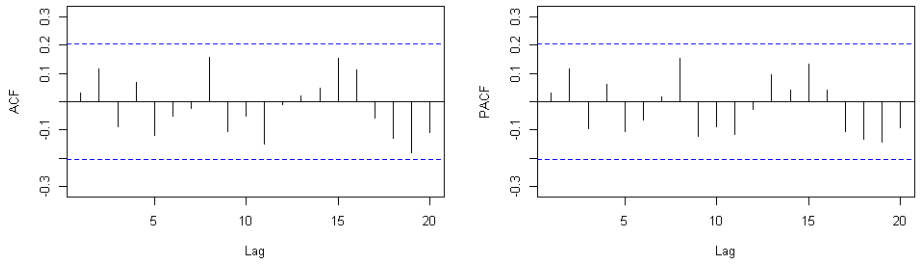

Figura 9.83: Residuos da série produção trimestral dos carros no Reino Unido utilizando o modelo auto-regressivo integrado de médias moveis sazonais

O modelo obteve um ajustamento através da análise do $MAPE = 6.43$, ou seja, o modelo teve um acerto de ajustamento de 93.6%, ilustrados na figura 9.84 respectivamente.

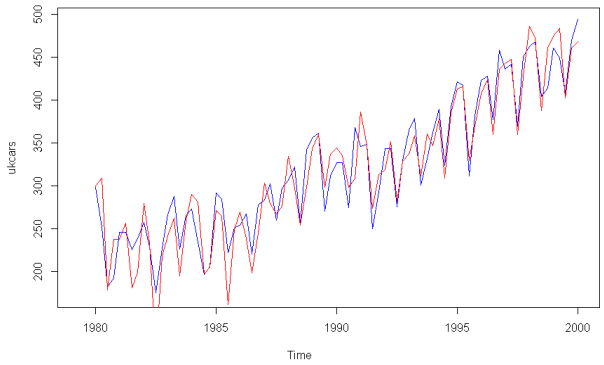

Figura 9.84: Ajustamento da série produção dos carros no Reino Unido utilizando o modelo auto-regressivo integrado de médias moveis sazonais

Verificando as redes neuronais com o algoritmo RNN para a série produção dos carros no Reino Unido, utilizaram-se os seguintes hiperparâmetros para o treinamento, a função de ativação o logistica, com 6 dimensões escondidas, para uma taxa de aprendizado de 80%, com 50 retropropagações, para um único lote de treinamento, resumida na tabela 9.33.

Hiperparâmetros	**Resultados**
network type	RNN
sigmoid	logistica
hidden-dim	6
learningrate	0.8
numepochs	50
batch size	1

Tabela 9.33: Hiperparâmetros do modelo da rede RNN ukcars

O erro do treinamento do modelo vai diminuindo a medida que se vão aumentando o número de épocas, ilustrado na figura 9.85, é de realçar as oscilações durante o treinamento até atingir o minimo.

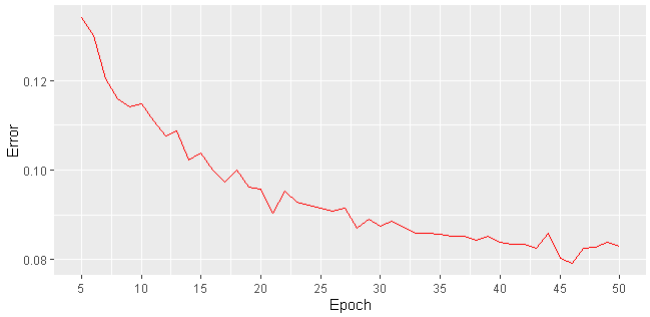

Figura 9.85: Evolução do erro do modelo utilizando o algoritmo RNN para a série produção trimestral dos carros no Reino Unido

Quanto ao ajustamento verifica-se que se vai melhorando mesmo que subtilmente com a época de treinamento. A acurácia de ajustamento foi com um $MAPE = 8.9$, ou seja, o modelo consegue com os hyperparâmetros fixados se ajustar a 91.1% aos dados, como mostra na figura 9.86.

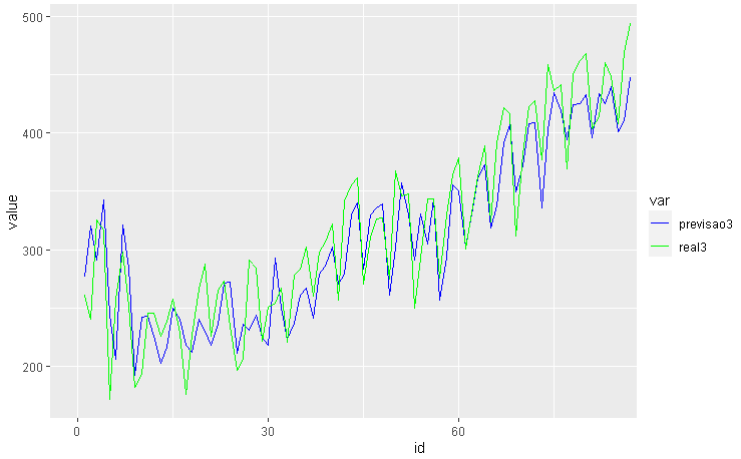

Figura 9.86: Ajustamento do modelo utilizando o algoritmo RNN para a série produção trimestral dos carros no Reino Unido

O algoritmo GRU, utilizando a função de ativação logistica, com uma taxa de aprendizado de 75%, com um treinamento com 70 retroprogações e um único lote, verificou-se também que tal como o algoritmo RNN, o erro vem diminuindo mas com oscilações. Os hiperparâmetros são resumidos na tabela 9.34.

Hiperparâmetros	**Resultados**
network type	GRU
sigmoid	logistica
hidden-dim	17
learningrate	0.75
numepochs	70
batch size	1

Tabela 9.34: Hiperparâmetros de treino do algoritmo GRU para a série produção trimestral dos carros no Reino Unido

O erro vai diminuindo a medida que se vai treinando, mas também com, neste caso, oscilações durante todo o treinamento, até a última época de treinamento, como mostra a figura 9.87.

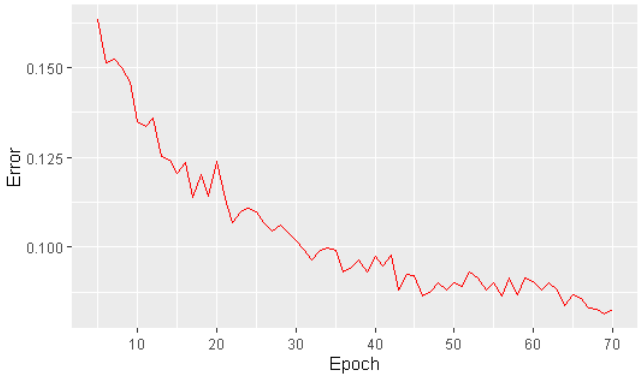

Figura 9.87: Evolução do erro por época modelo utilizando o algoritmo GRU para a série produção dos carros no Reino Unido

Quanto ao ajustamento, obteve um MAPE = 9.2, ou seja, o algoritmo teve um acerto no ajustamento de 90.8%, ilustrada na figura 9.88.

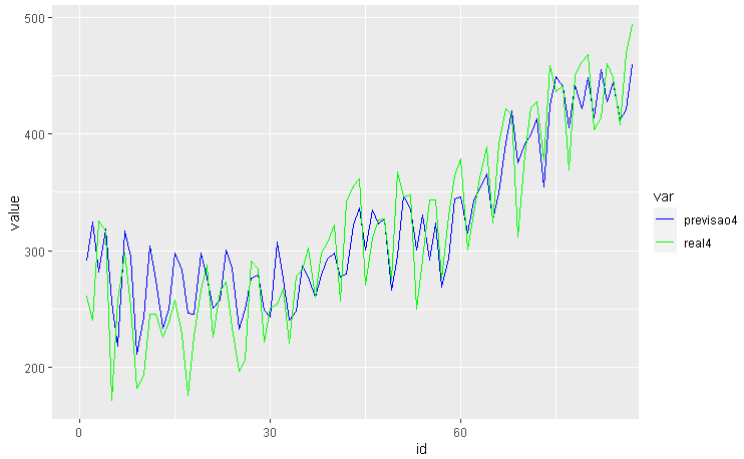

Figura 9.88: Ajustamento utilizando o algoritmo GRU para a série produção dos carros no Reino Unido

O algoritmo LSTM, comparativamente com os outros algoritmos supervisionado de aprendizado de máquina, obteve o pior resultado, apesar erro tem uma descida gradual e constante sem grandes oscilações, até atingir o minimo para série de treinamento, ilustrada na figura 9.89.

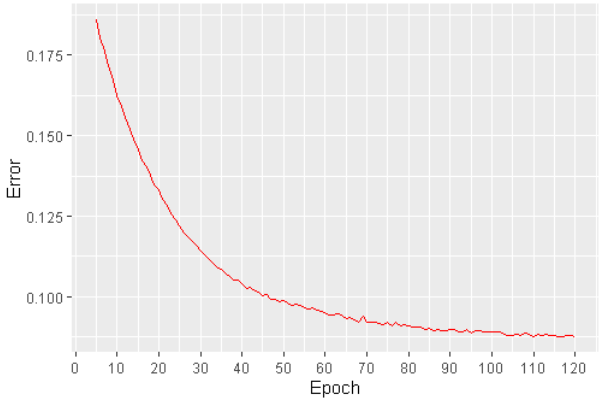

Figura 9.89: Evolução do erro por época utilizando o algoritmo LSTM para a série produção trimestral dos carros no Reino Unido

Quanto aos hiperparâmetros utilizados, foram a função de ativação, uma tangente hiperbolica, com uma taxa de aprendizado por época de 0.85, para os 120 retropropagações efetuadas, e utilizou-se um único lote de treinamento, resumida na tabela 9.35.

Hiperparâmetros	**Resultados**
network type	LSTM
sigmoid	tanh
hidden-dim	14
learningrate	0.85
numepochs	120
batch size	1

Tabela 9.35: Hiperparâmetros do modelo da rede LSTM para a série produção trimestral dos carros no Reino Unido

Quando ao ajustamento, obteve um MAPE = 11.05, ou seja, o algoritmo teve um acerto no ajustamento de 89%, ilustrada na figura 9.90.

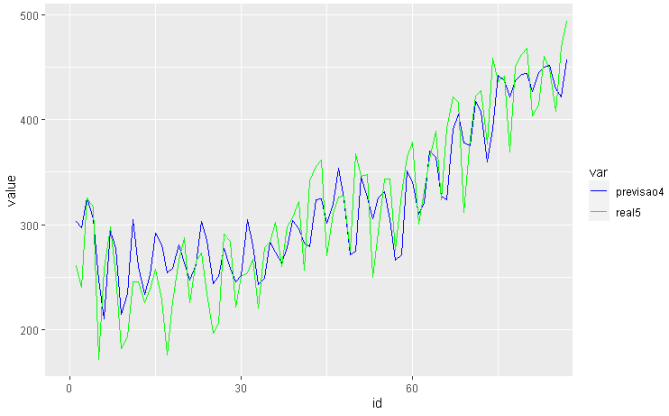

Figura 9.90: Ajustamento do modelo utilizando o algoritmo LSTM para a série produção trimestral dos carros no Reino Unido

Quanto aos resultados e resumido na tabela 9.24 utilizando os diferentes métodos de ajustamento utilizado constata-se que o método estatistico através do modelo SARIMA $(3,1,2)(1,1,0)_{12}$ tendo a melhor performance de ajustamento. Constata-se que muito próximo estão as redes com o algoritmo RNN, GRU e LSTM que tiveram desempenhos próximos.

	ME	**RMSE**	**MAE**	**MPE**	**MAPE**
ARIMA	-2.77	23.38	17.24	-1.97	6.43
RNN	1.28	31.55	25.51	0.92	8.90
GRU	-0.79	31.80	26.14	-1.93	9.20
LSTM	-2.83	38.67	31.55	-3.16	11.05

Tabela 9.36: Resumo dos resultados dos diferentes modelos para a série produção trimestral dos carros no Reino Unido

10

Considerações finais e limitações

A análise de séries temporais tem evoluído ao longo do tempo, onde inicialmente se utilizavam simples decomposições para a compreensão da sequência dos dados, e todas as suas oscilações, passou-se pelos métodos clássicos e posteriormente pelos métodos lineares, para hoje chegarmos ao estágio mais moderno dessa análise utilizando as técnicas de inteligência artificial para essa compreensão, este que consiste num mecanismo de treino de algoritmos, para um aprendizado dos padrões dos dados. O papel da dimensão dos dados (número de observações da série) se revelou, nesse caso, fundamental nessa análise.

Concluiu-se que a utilização de técnicas, ou modelos, estatísticos (modelos ARIMA, entre outros) ou da inteligência artificial para a análise de dados sequênciais requer um conhecimento dos dados para uma utilização apropriada dessas técnicas; esse conhecimento passa por saber se a série é uma série longa ou curta, se a série é estacionária ou se possui tendência e se essa tendência é crescente ou decrescente, também verificar se a série comporta períodos sazonais, ou possui tanto tendência como sazonalidade.

Durante esse estudo realizado verificou-se que a dimensão das séries foi fundamental para o resultado alcançado. As técnicas estatísticas se revelaram melhores nos ajustamentos realizados, isso tanto para dados não estacionários com tendência, como para dados não estacionários com tendência e sazonalidade, obtendo resultados melhores utilizando as diferentes medidas de acurácia. Esse maior desempenho das

técnicas estatísticas se deve também fortemente ao tempo inicial de aprendizado das redes, em que no início as redes têm maior dificuldade de ajustamento, enquanto que as técnicas estatísticas começam logo de início com um bom ajustamento.

Verificou-se também que apesar de o desempenho inicial não ser o melhor, as técnicas de inteligência artificial são capazes de compreender muito bem o padrão dos dados, mesmo quando existem movimentos bruscos na trajetória da série.

Entre as técnicas modernas de inteligência artificial para dados sequênciais o algoritmo das redes neuronais artificiais (RNN) obteve o melhor resultado, seguido do algoritmo de memória de Longo Prazo (LSTM) e do algoritmo com unidades recorrentes com portão (GRU). Isso se deve ao fato de que os algoritmos LSTM e GRU são projetados especificamente para séries longas, fazendo com que o seu desempenho seja menor em series curtas, diferente do algoritmo RNN, de modo que em muitos casos o seu desempenho seja muito próximo das técnicas estatísticas.

10.1 Limitações do trabalho

A utilização de técnicas de inteligência artificial requer uma memória computacional considerável, isso torna mais importante quando a utilização de inteligência artificial, nomeadamente as redes neuronais se desenrola primeiramente no *software* python, e para a sua utilização no *software* R, obriga a uma compatibilidade e instalação dos dois *softwares*. Essa impossibilidade disponível limitou a utilização de série temporais mais longas, impossibilitando uma mesma comparação para série longas.

10.2 Trabalhos futuros

Para os trabalhos futuros, torna-se uma condição necessária trabalhar com série longas, para novas comparações. Também propõem-se uma comparação dos dados a partir de alguns períodos a frente (onde temos um melhor ajustamento) para uma comparação equitativa em virtude do tempo de aprendizagem.

Também será interessante avaliar-se não só a performance das metodologias clássicas (ou estatísticas) e das técnicas de inteligência artificial não só no que diz respeito ao ajustamento mas também no que diz respeito à previsão de valores futuros. Esta abordagem constava dos objectivos iniciais desta dissertação mas devido à morosidade, e novidade, na implementação das técnicas de inteligência artificial tal não foi possível de realizar. Considera-se que em determinados casos as técnicas de inteligência artificial poderão ser substancialmente melhores a prever do que as metodologias clássicas (ou estatísticas).

Uma outra perspectiva de trabalho e a utilização de séries não lineares, tanto para série longas ou curtas, estacionárias ou não estacionárias, para um resultado total para todos os tipos de dados sequenciais.

Bibliografia

[1] Bianchi, Filippo M.; Maiorino, Enrico; Kampffmeyer, Michael C.; Rizzi,; Antonello; Jenssen, Robert, *Recurrent Neural Networks for Short-Term Load Forecasting: An Overview and Comparative Analysis*, SpringerBriefs in Computer Science, 2017.

[2] Box, George E. P.; Jenkins, Gwilym M.; Reinsel, Gregory C.; Ljung, Greta M., *TIME SERIES ANALYSIS: Forecasting and Control*, Fifth Edition, John Wiley & Sons, Inc.,New Jersey, 2016.

[3] Braga, Antônio P.; Carvalho André P. L.; Ludermir, Teresa B., *Redes Neurais Artificiais: Teoria e Aplicações*, Rio de Janeiro - CEP 20040-040, LTC - Livros Técnicos e Científicos Editora S.A., 2000.

[4] Bueno, Rodrigo, *Econometria de séries temporais*, 2º edição, Cengage Learning, São Paulo, 2012.

[5] Cameron, A. Colin; Trivedi, Pravin K., *Microeconometrics: Methods and Applications*, Cambridge University Press, New York, 2005.

[6] Ciaburro, Giuseppe; Venkateswaran, Balaji, *Neural Networks with R*, Birmingham, Packt Publishing Ltd., 2017.

[7] Chatfield, Chris, *Time-Series Forecasting*, Chapman & CHall/CRC, New York,2000.

[8] Chollet, Francois ; Allaire, J.J, *Deep Leaning with R*, MEAP-Edition Manning Early Access Program, 1º edition, California, 2018.

[9] Christodoulakis, George; Satchell, Stephen, *The Analytics of Risk Model Validation*, 1º edition, Elsevier Ltd., Massachusetts, 2008.

[10] Cowpertwait, Paul S.P.; Metcalfe, Andrew V., *Introductory Time Series with R*, Springer Science+Business Media, New York, 2009.

[11] Cryer, Jonathan D.; Chan, Kung-Sik, *Time Series Analysis With Applications in R*, Second Edition, Springer Science+Business Media, LLC, New York, 2008.

[12] Daróczi, Gergely; Puhle, Michael; Berlinger, Edina; Csóka, Péter; Havran, Dániel; Michaletzky, Márton; Tulassay, Zsolt ; Váradi, Kata ; Vidovics-Dancs, Agnes, *Introduction to R for Quantitative Finance*, Packt Publishing Ltd., Birmingham , 2013.

[13] Derryberry, DeWayne R., *Basic Data Analysis for Time Series with R*, John Wiley & Sons, Inc., Hoboken, New Jersey, 2014.

[14] Faraway, Julian J., *Linear Models with R*, Chapman & Hall/CRC, Massachusetts, USA, 2009.

[15] Ferreira, Pedro; Mattos, Daiane; Almeida, Diego; Oliveira, Ingrid; Pereira, Rafael; *Análise de Séries Temporais em R: um curso introdutório*, Instituto Brasileiro de Economia, 2017.

[16] Ghatak, Abhijit, *Deep Learning with R*, Singapore, Springer Nature Singapore Pte Ltd., 2019.

Goldberg, Yoav, *Neural Network Methods for Natural Language Processing*, Morgan & Claypool Publishers series, Canada, 2017.

[17] Greene, William H., *Econometric Analysis*, Fifth Edition, Prentice Hall, New Jersey, 2003.

[18] Hammer, Barbara, *Learning with Recurrent Neural Networks* , Great Britain, Springer-Verlag London, 2000.

[19] Harvey, Robert L., *Neural Network Principles*, Massachusetts, Prentice-Hall International-Inc., 1994.

[20] Haykin, Simon, *Redes Neurais: Principios e prática*, 2º edição, Prentice Hall,Inc., Artmed Editora S.A, Brasil, 1999.

[21] Hyndman, Rob J.; Koehler, Anne B.; Ord, J. Keith ; Snyder, Ralph D., *Forecasting with Exponential Smoothing: The State Space Approach*, Springer-Verlag Berlin Heidelberg, Berlin - Germany, 2008.

[22] Kendall, Maurice; Ord, Keith, *Times series*, third edition, Hodder Arnold, Reino Unido, 1993.

[23] Kirchgässner, Gebhard; Wolters, Jürgen, *Introduction to Modern Time Series Analysis*, Springer-Verlag Berlin Heidelberg, Leipzig - Germany, 2007.

[24] Kleiber, Christian; Achim Zeileis, *Applied Econometrics with R*, Baltimore-USA, Springer Science+Business Media, LLC, 2008.

[25] McLeod, A. Ian; Yu, Hao; Mahdi, Esam, *Time Series Analysis with R*, Elsevier, Ontario, 2011.

[26] Morettin, Pedro; Toloi, Clelia, *Análise de séries temporais*, 2º edição, Blucher, São Paulo, 2006.

[27] Morettin, Pedro, *Econometria financeira: Um curso em séries temporais financeiras*, 2º edição, Blucher, São Paulo, 2006.

[28] Levy, George, *Computational Finance: Numerical Methodsfor Pricing Financial Instruments*, Elsevier Butterworth-Heinemann, Burlington, MA, 2004.

[29] Palit, Ajoy K.; Popovic Dobrivoje, *Intelligence in Time Series Forecasting: Theory and Engineering Applications*, Springer, London, 2005.

[30] Planas, Christophe, *Applied Time Series Analysis: Modelling, Forecasting, Unobserved Components Analysis, and the Wiener-Kolmogorov Filter*, Office for Official Publications of the European Communities, Luxembourg, 1997.

[31] Prado, Raquel; West, Mike; *Time Series Modeling, Computation and Inference*, Taylor and Francis Group, Boca Raton, FL, 2010.

[32] Prakash, PKS; Rao, Achyutuni S. K., *R Deep Learning: Solve complex neural net problems with TensorFlow, H20 and MXNet*, Packt Publishing Ltd., Birmingham, 2017.

[33] Programmer, Lazy, *Deep Learning: Recurrent Neural Networks in Python, LSTM, GRU, and more RNN machine learning architectures in Python and Theano*, Kindle, United States of America, 2016.

[34] Rao, A. Ravishankar; Cecchi, Guillermo A., *The Relevance of the Time Domain to Neural Network Models*, Volume 3, Springer Science+Business Media, LLC, , New York, 2012.

[35] Rashid, Tariq, *Make Your Own Neural Network*, Kindle, England, 2016

[36] Rzempoluck, Edward J., *NEURAL NETWORK DATA ANALYSIS USING SIMULNET*, 1st edition, Springer Science+Business Media, New York, 1998.

[37] Shmueli, Galit; Lichtendahl, Kenneth C. Jr., *PRACTICAL TIME SERIES FORECASTING WITH R*, 2º edition, Axelrod schnall, Florida, 2016.

[38] Shumway, Robert H.; Stoffer, David S., *Time Series Analysis and Its Applications: With R Examples*, 3º edition, Springer Science+Business Media, New York, 2010.

[39] Steiner, Robert, *Mastering Financial Calculations: A step-by-step guide to the mathematics of financial market instruments*, Pearson Education, Great Britain, 1998.

[40] Temple, Peter, *First steps in Economic Indicators*, Pearson Education, Great Britain, 2003.

[41] Warner, Stuart & Hussain, Si, *The Finance Book*, First edition, Pearson Education, United Kingdom, 2017.

[42] Wickham, Hadley, *ggplot2: Elegant Graphics for Data Analysis*, Springer Science+Business Media, LLC, New York - USA, 2009.

[43] Wilde, Philippe De, *Neural Network Models: An Analysis*, Springer-Verlag, Great Britain, 1996

[44] Wooldridge, Jeffrey M., *Introductory econometrics: a modern approch*, 4º edicao, Thomson Learning, Inc., Michigan, 2006.

[45] YI, Zhang; TAN, K.K., *Network Theory and Applications: CONVERGENCE ANALYSIS OF RECURRENT NEURAL NETWORKS*, 1st edition, Springer-Science+Business Media, B.V., New York, 2004.

[46] <https://iaexpert.academy/2020/06/08/os-tipos-de-redes-neurais/>. Acesso em: 12 de dez. de 2019

[47] <http://deeplearningbook.com.br/as-10-principais-arquiteturas-de-redes-neurais/>. Acesso em: 15 de jan. de 2020.

[48] <https://otexts.com/fpp2/> Acesso em: 15 de jan. de 2020.

[49] <https://caiquecoelho.medium.com/um-guia-completo-para-o-pre-processamento-de-dados-em-machine-learning-f860fbadabe1>. Acesso em: 7 de abr. de 2020.

[50] <https://www.monolitonimbus.com.br/redes-neurais-artificiais/>. Acesso em: 20 de jun. de 2020.

[51] <https://keras.rstudio.com/reference/layer-activation-leaky-relu.html>. Acesso em: 20 de jun. de 2020.

[52] <https://stackoverflow.com/questions/34518656/how-to-interpret-loss-and-accuracy-for-a-machine-learning-model>. Acesso em: 1 de ago. de 2020.

[53] <https://www.cetax.com.br/blog/compreendendo-lstm-networks/>. Acesso em: 15 de ago. de 2020.

[54] <https://www.mariofilho.com/as-metricas-mais-populares-para-avaliar-modelos-de-machine-learning/>. Acesso em: 1 de set. de 2020

11

Anexo

11.1 Série morte Estados Unidos

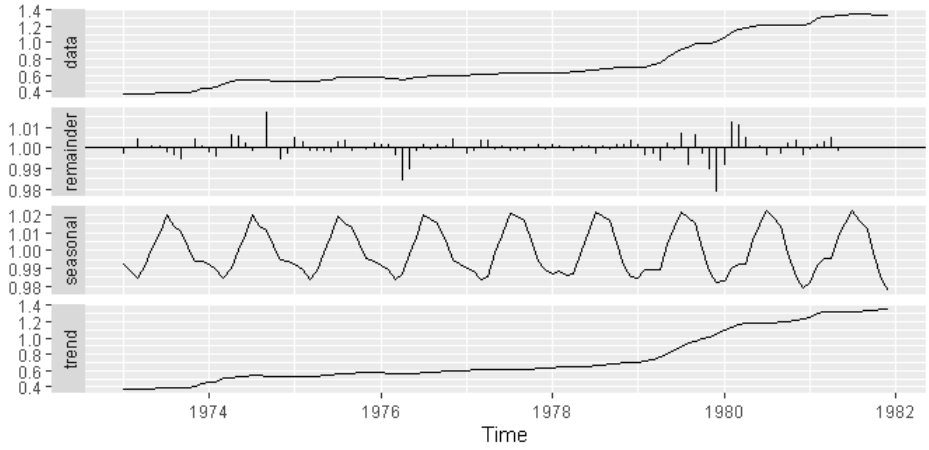

Figura 11.1: Decomposição da série morte nos Estados Unidos utilizando o método X11

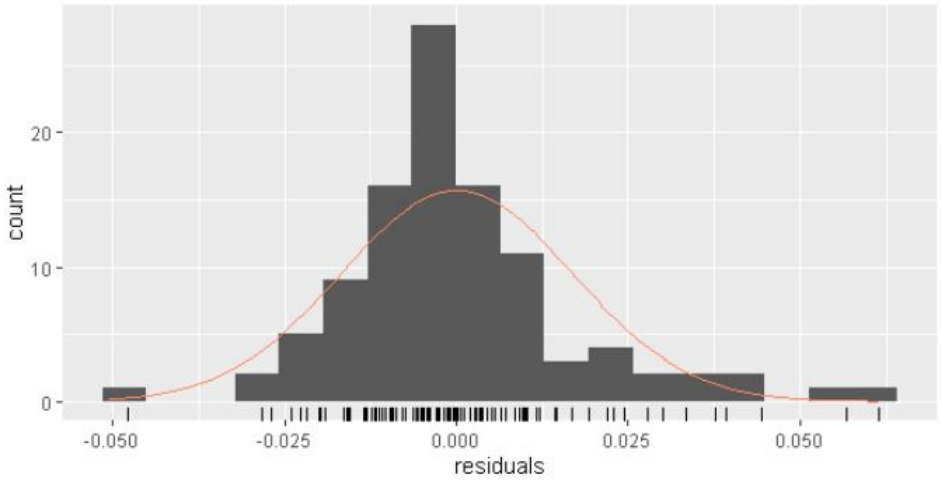

Figura 11.2: Residuos do modelo da série morte nos Estados Unidos

Diversos outros metodos de análise de séries temporais são apresentados na tabela 11.1, onde se verifica que alguns modelos tiveram um bom desempenho, como splinef com um MAPE de 7.48, o modelo stlm-arima com um MAPE de 7.45, o modelo StructTS com um MAPE de 7.08, e o modelo stlmets tendo sido o melhor modelo, para a série morte nos Estados Unidos com um MAPE de 4.11. Outros modelos como o meanf e o croston com um MAPE de 52.18 e 33.11, tiveram os piores desempenhos.

$\underline{model}$	$\underline{RMSE}$	$\underline{MPE}$	$\underline{MAPE}$	$\underline{MASE}$	$\underline{melhor-modelo}$	$\underline{runtime-model}$
auto.arima	0.13062833	9.989009	9.989009	1.5787169	stlm-ets	33.52
bats	0.15707494	11.833637	11.833637	1.8793316	stlm-ets	2.55
croston	0.42554481	33.112659	33.112659	5.2249445	stlm-ets	0.43
ets	0.20524892	15.160225	15.160225	2.4178482	stlm-ets	1.70
hybrid	0.17340647	13.132495	13.132495	2.0839861	stlm-ets	6.84
meanf	0.66067317	52.177092	52.177092	8.1962612	stlm-ets	0.00
naive	0.25402510	18.943965	18.943965	3.0166609	stlm-ets	0.00
nnetar	0.14913014	10.069252	10.069252	1.6228229	stlm-ets	0.72
rwf	0.25402510	18.943965	18.943965	3.0166609	stlm-ets	0.00
rwf-drift	0.14898692	11.449017	11.449017	1.8084960	stlm-ets	0.01
snaive	0.40419550	30.810674	30.810674	4.8316557	stlm-ets	0.00
splinef	0.09840736	7.482580	7.482580	1.1686536	stlm-ets	0.04
stlm-arima	0.09839070	7.447652	7.447652	1.1694776	stlm-ets	0.25
stlm-ets	0.06113475	2.473184	4.113605	0.6337906	stlm-ets	0.04
StructTS	0.09380889	7.078727	7.078727	1.1036601	stlm-ets	0.38
tbats	0.21106557	15.623476	15.623476	0.8277368	stlm-ets	5.58
thetaf	0.21660844	16.318384	16.318384	0.7891722	stlm-ets	0.00
tslm	0.37020393	29.250673	29.250673	0.7462327	stlm-ets	0.02

Tabela 11.1: Resumo do desempenho dos diferentes modelos utilizando diferentes métricas da série morte Estados Unidos

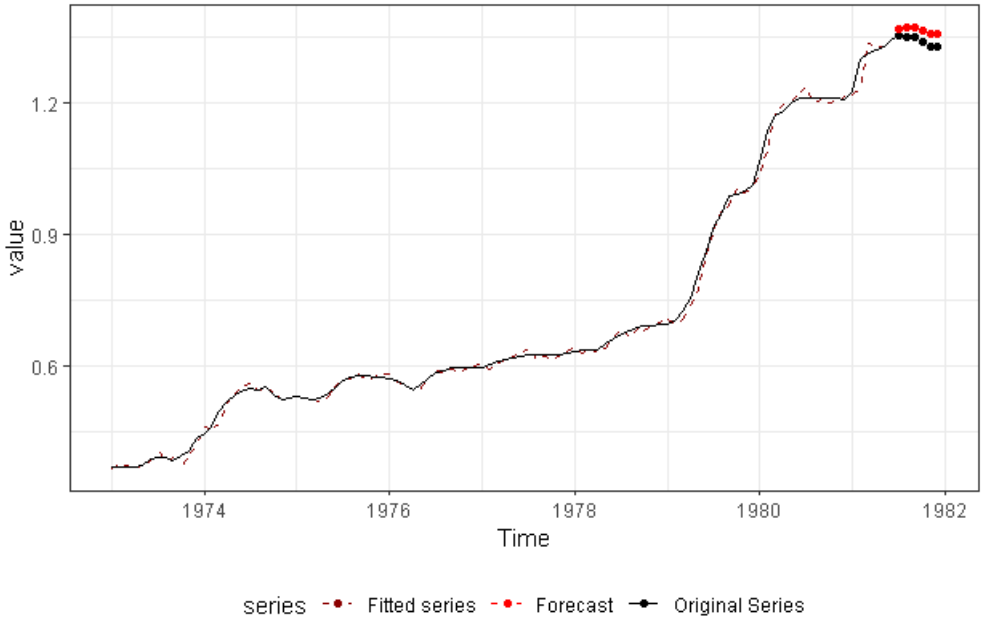

Figura 11.3: Gráfico da série morte nos Estados Unidos utilizando o método stlm-ets

11.2 Série consumo público em Portugal

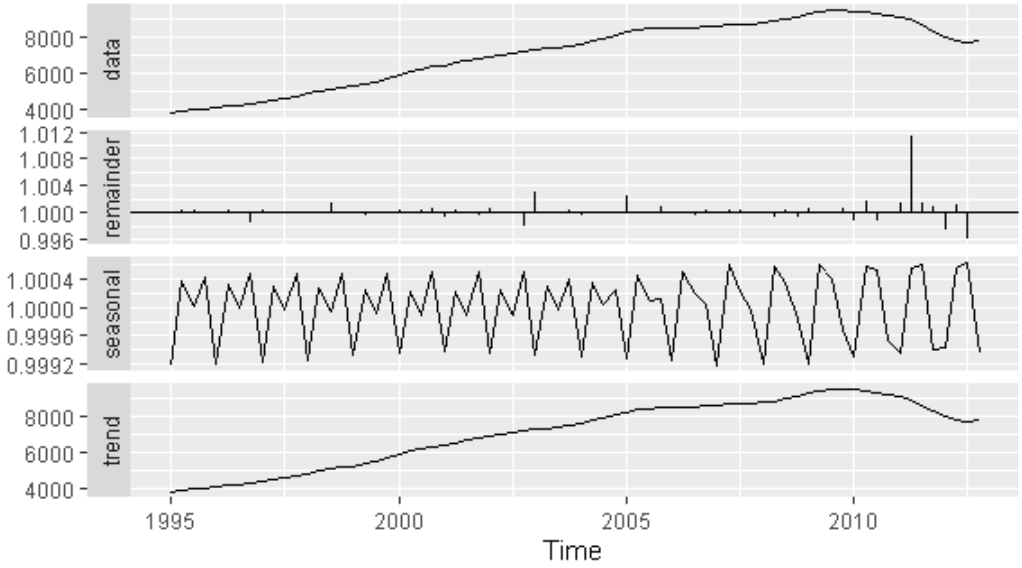

Figura 11.4: Decomposição da série Consumo público utilizando o metodo X11

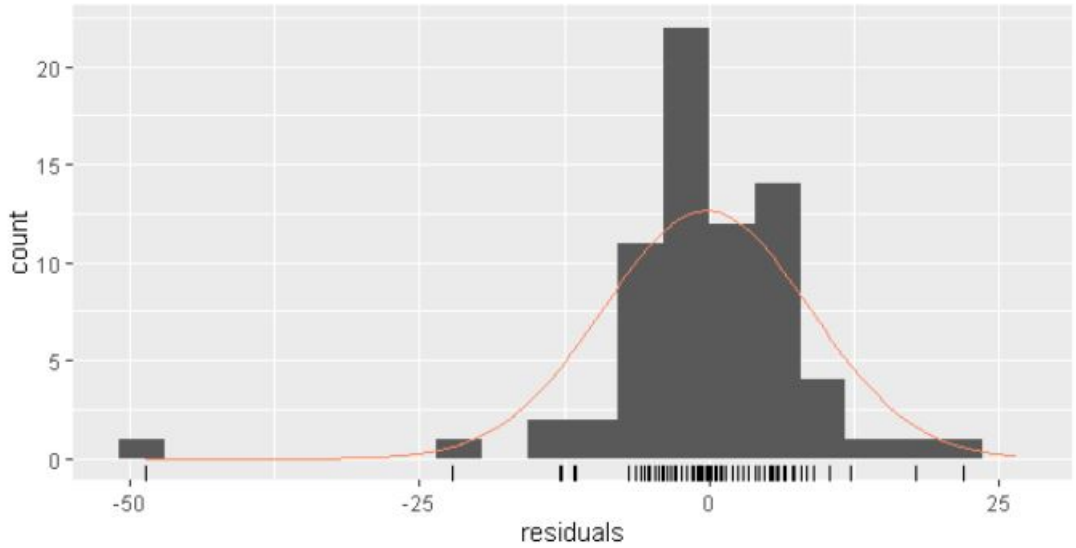

Figura 11.5: Residuos do modelo da série consumo público

Quanto a série Consumo Público os resultados são apresentados na tabela 11.2, onde se verifica que globalmente todos os modelos tiveram um bom desempenho a exceção do modelo meanf (MAPE = 14.2), e do modelo tslm (MAPE = 16.6). O melhor modelo encontrado foi um rwf-drift (MAPE = 0.22).

$model$	$RMSE$	MPE	$MAPE$	$MASE$	$melhor-modelo$	$runtime-model$
auto.arima	106.22692	1.05888771	1.0588877	0.23682149	rwf-drift	1.58
bats	108.46207	1.07256972	1.0725697	0.23992697	rwf-drift	0.92
croston	253.99156	2.78835486	2.7883549	0.62212691	rwf-drift	0.56
ets	75.77368	0.73908486	0.7390849	0.16531882	rwf-drift	0.40
hybrid	127.27923	1.31151673	1.3115167	0.29310098	rwf-drift	2.14
meanf	1207.80559	14.21369804	14.2136980	3.16076877	rwf-drift	0.00
naive	204.29012	2.14757071	2.1475707	0.47974857	rwf-drift	0.00
nnetar	474.01394	-5.00582854	5.0058285	1.11531185	rwf-drift	0.80
rwf	204.29012	2.14757071	2.1475707	0.47974857	rwf-drift	0.00
rwf-drift	24.93391	-0.03143373	0.2228738	0.04932413	rwf-drift	0.00
snaive	267.46845	2.92476474	2.9247647	0.65243705	rwf-drift	0.00
splinef	68.84846	0.68798441	0.6879844	0.15377092	rwf-drift	0.04
stlm-arima	107.23293	1.06690355	1.0669036	0.23862894	rwf-drift	0.16
stlm-ets	109.91362	1.08591906	1.0859191	0.24292261	rwf-drift	0.03
StructTS	88.68864	0.91831021	0.9183102	0.20517626	rwf-drift	0.03
tbats	106.80550	1.05554982	1.0555498	0.23612090	rwf-drift	2.44
thetaf	107.38362	1.13140698	1.1314070	0.25265729	rwf-drift	0.02
tslm	1404.28454	-16.58428339	16.5842834	3.68511789	rwf-drift	0.02

Tabela 11.2: Resumo do desempenho dos diferentes modelos utilizando diferentes métricas da série consumo publico

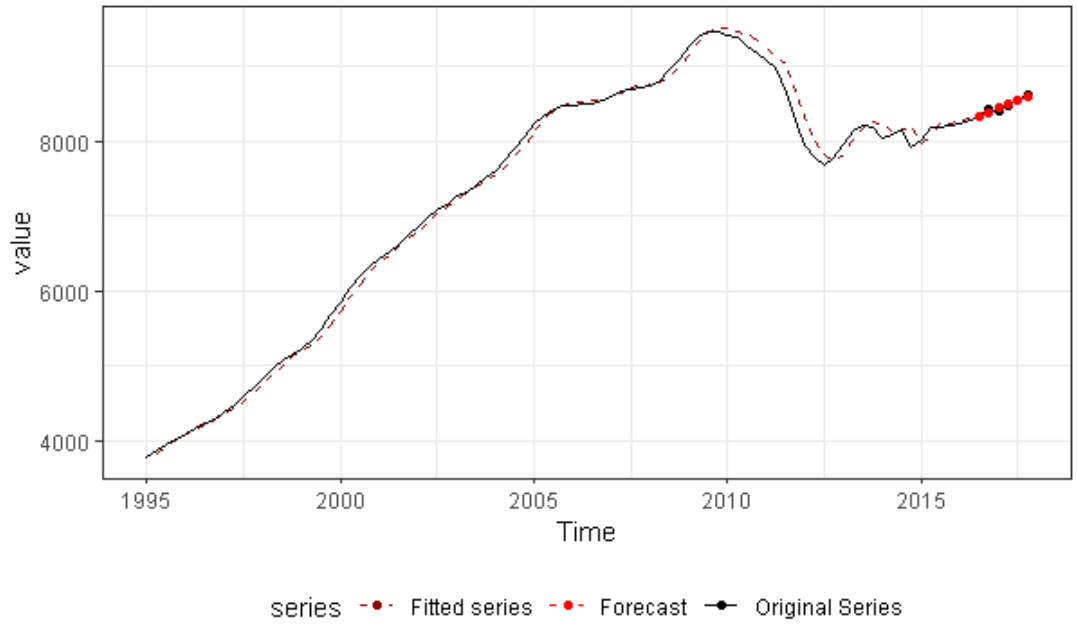

Figura 11.6: Gráfico da série consumo público utilizando o método rwf-drift

11.3 Série produto interno bruto em Portugal

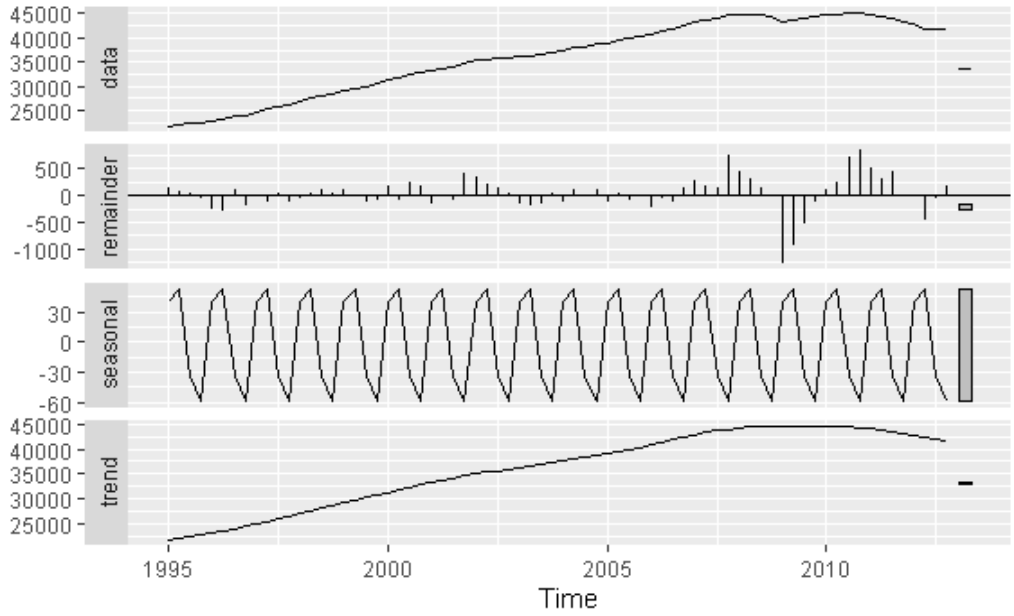

Figura 11.7: Decomposição da série Produto Interno Bruto utilizando o método STL

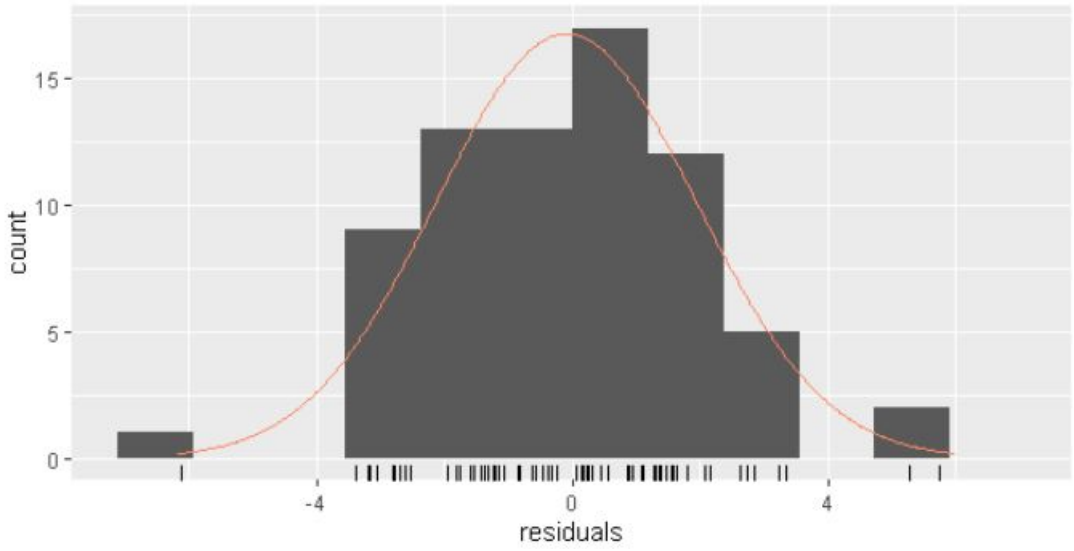

Figura 11.8: Residuos do modelo da série PIB

Analisando a série Produto Interno Bruto, cujo o resultado se apresenta na tabela 11.3, onde se verifica que globalmente todos os modelos tiveram um bom desempenho a exceção do modelo meanf (MAPE = 14.2), e do modelo tslm (MAPE = 16.6). O melhor modelo encontrado foi um rwf-drift (MAPE = 0.22).

$model$	$RMSE$	MPE	$MAPE$	$MASE$	$melhor-modelo$	$runtime-model$
auto.arima	1159.9943	2.184149	2.184149	0.6908961	stlm-arima	2.31
bats	1205.2283	2.281582	2.281582	0.7215292	stlm-arima	1.64
croston	3963.6477	8.004126	8.004126	2.5218038	stlm-arima	0.37
ets	910.5806	1.753501	1.753501	0.5540647	stlm-arima	0.35
hybrid	1428.6611	2.685747	2.685747	0.8496266	stlm-arima	3.08
meanf	10750.8961	22.230899	22.230899	6.9861558	stlm-arima	0.00
naive	2142.8092	4.041073	4.041073	1.2782005	stlm-arima	0.00
nnetar	4051.3297	7.538892	7.538892	2.3860031	stlm-arima	0.95
rwf	2142.8092	4.041073	4.041073	1.2782005	stlm-arima	0.00
rwf-drift	1028.5393	1.975247	1.975247	0.6242202	stlm-arima	0.00
snaive	2748.4337	5.348689	5.348689	1.6880705	stlm-arima	0.02
splinef	815.6225	1.565297	1.565297	0.4946806	stlm-arima	0.31
stlm-arima	742.7412	1.413927	1.413927	0.4469605	stlm-arima	0.05
stlm-ets	773.0359	1.470279	1.470279	0.4647997	stlm-arima	0.03
StructTS	884.9545	1.699898	1.699898	0.5371946	stlm-arima	0.04
tbats	1393.7814	2.563990	2.563990	0.8118749	stlm-arima	3.97
thetaf	1601.7965	3.040705	3.040705	0.9614814	stlm-arima	0.02
tslm	2230.1105	-4.576115	4.576115	1.4314962	stlm-arima	0.00

Tabela 11.3: Resumo do desempenho dos modelos utilizando diferentes métricas da série produto interno bruto

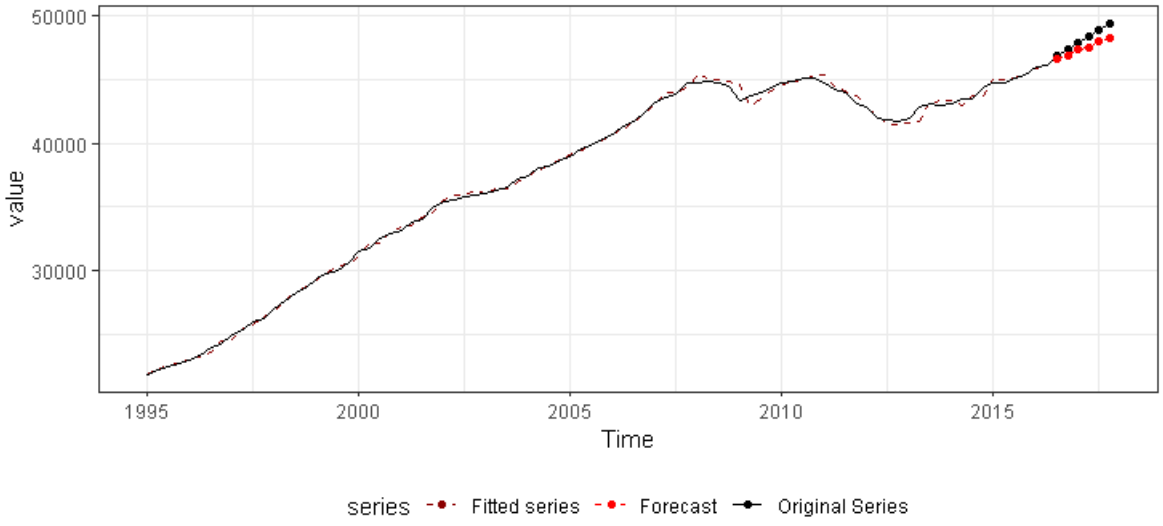

Figura 11.9: Gráfico da série Produto interno bruto utilizando o método stlm-arima

11.4 Série temperatura do corpo de castor

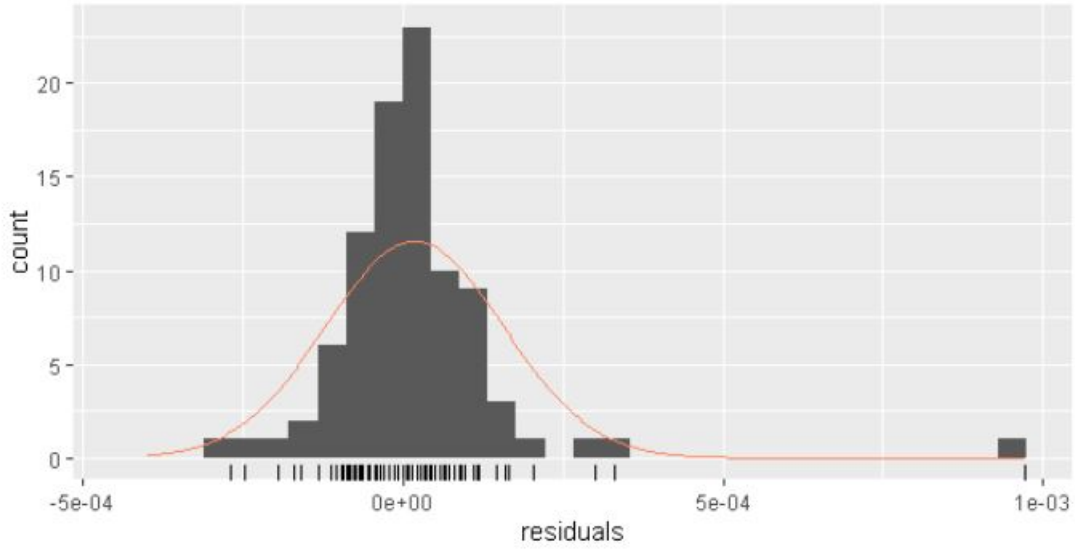

Figura 11.10: Residuos do modelo da série Castor

O resume do resultado da série Castor2, se apresenta na tabela 11.4, onde se verifica que assim como a série anterior, os resultados são globalmente bons em todos os modelos, mas o modelo thetaf obteve o melhor resultado (MAPE =0.44), e embora tendo sido o modelo com pior desempenho o modelo splinef (MAPE = 1.68), pode-se considerar que o resultado é satisfatório.

$\underline{model}$	$\underline{RMSE}$	$\underline{MPE}$	$\underline{MAPE}$	$\underline{MASE}$	$\underline{melhor-modelo}$	$\underline{runtime-model}$
auto.arima	0.2239606	0.03405775	0.5055186	1.976488	thetaf	0.93
bats	0.2239486	0.03355385	0.5054371	1.976160	thetaf	1.57
croston	0.2716172	-0.41273540	0.6381375	2.486569	thetaf	0.52
ets	0.2239609	0.03407384	0.5055212	1.976499	thetaf	0.03
hybrid	0.2088854	-0.09998263	0.4547642	1.775740	thetaf	1.06
meanf	0.2603379	0.35044381	0.5566676	2.182454	thetaf	0.00
naive	0.2239606	0.03405775	0.5055186	1.976488	thetaf	0.00
nnetar	0.2812359	-0.21450996	0.6560536	2.564917	thetaf	0.81
rwf	0.2239606	0.03405775	0.5055186	1.976488	thetaf	0.00
rwf-drift	0.1940012	-0.18715298	0.4479073	1.747321	thetaf	0.02
snaive	0.2239606	0.03405775	0.5055186	1.976488	thetaf	0.00
splinef	0.7912500	1.60255873	1.6822297	6.599210	thetaf	0.06
StructTS	0.1944469	-0.17742941	0.4463650	1.741461	thetaf	0.01
tbats	0.2239486	0.03355385	0.5054371	1.976160	thetaf	5.51
thetaf	0.2025054	-0.08837104	0.4424968	1.727898	thetaf	0.00

Tabela 11.4: Resumo do desempenho dos modelos utilizando diferentes métricas da série castor2

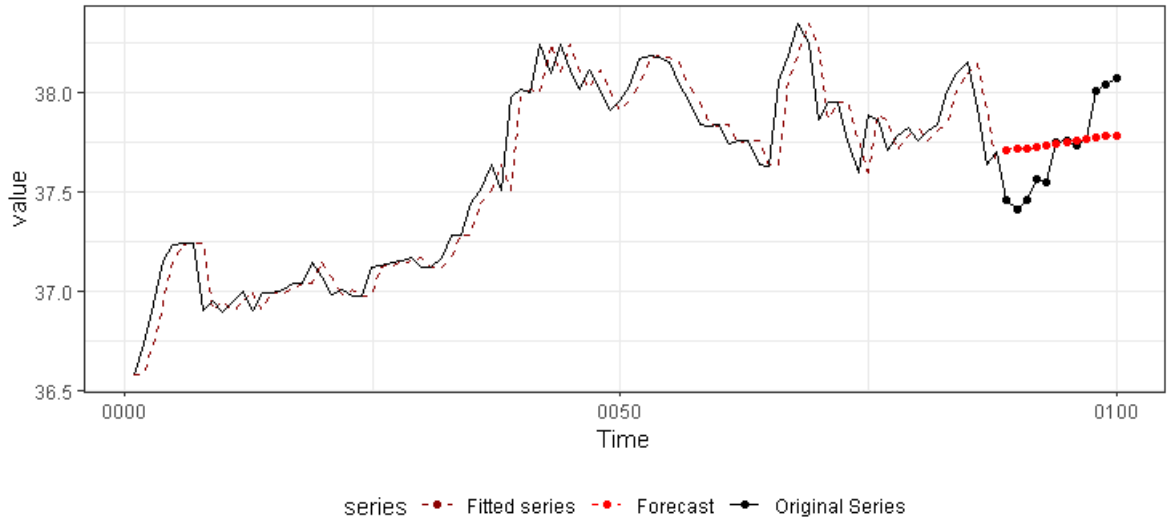

Figura 11.11: Gráfico da série Castor2 utilizando o método thetaf

11.5 Série preço de uma dúzia de ovos nos Estados Unidos

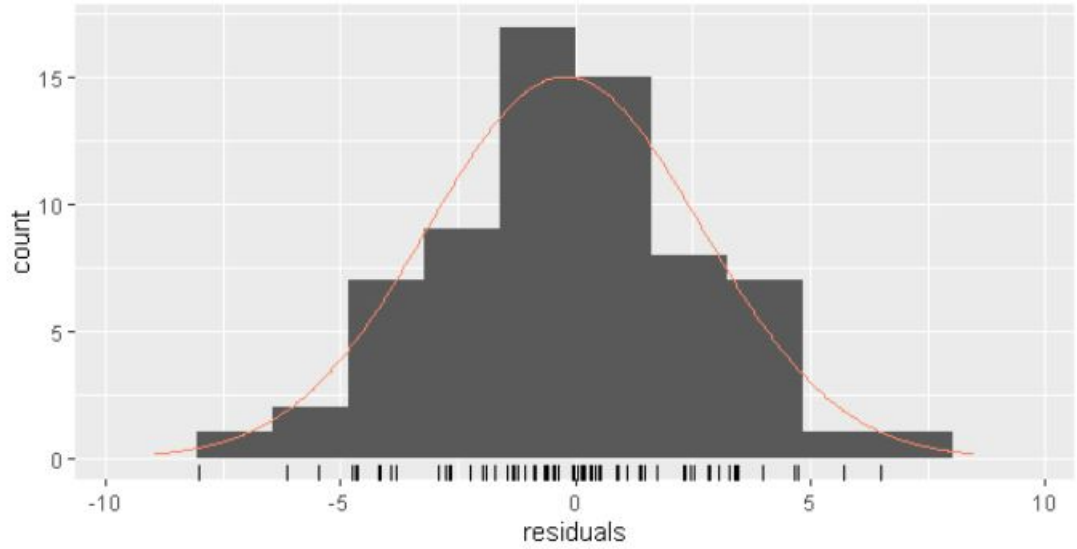

Figura 11.12: Residuos do modelo da série ovos

Quanto a série Eggs nos Estados Unidos, tabela 11.5 também obteve um bom desempenho verificando as diversas medidas de acurácia, onde o modelo rwf-drift (MAPE = 0.72), foi o melhor modelo encontrado, enquanto que o modelo meanf (MAPE = 6.47), foi o que teve a menor acurácia.

$model$	$RMSE$	MPE	$MAPE$	$MASE$	$melhor-modelo$	$runtime-model$
auto.arima	35.87843	-43.63002	43.67023	1.4408793	rwf-drift	0.66
bats	38.60777	-47.33707	47.33707	1.5708756	rwf-drift	1.05
croston	71.51766	-90.00344	90.00344	3.0908199	rwf-drift	0.30
ets	40.40909	-49.75698	49.75698	1.6570822	rwf-drift	0.03
hybrid	42.75398	-52.64507	52.64507	1.7546837	rwf-drift	0.68
meanf	147.09342	-184.84524	184.84524	6.4694585	rwf-drift	0.00
naive	35.87843	-43.63002	43.67023	1.4408793	rwf-drift	0.00
nnetar	128.97451	-148.18725	153.50379	5.1481242	rwf-drift	0.62
rwf	35.87843	-43.63002	43.67023	1.4408793	rwf-drift	0.02
rwf-drift	18.18792	-21.44504	21.72246	0.7226062	rwf-drift	0.00
snaive	35.87843	-43.63002	43.67023	1.4408793	rwf-drift	0.00
splinef	106.12963	122.21022	122.21022	4.0162822	rwf-drift	0.03
StructTS	21.75179	-26.27671	26.27671	0.8752163	rwf-drift	0.00
tbats	38.60777	-47.33707	47.33707	1.5708756	rwf-drift	2.03
thetaf	28.34414	-34.67261	34.67261	1.1526610	rwf-drift	0.00

Tabela 11.5: Resumo do desempenho dos diferentes modelos utilizando diferentes métricas da série Eggs Estados Unidos

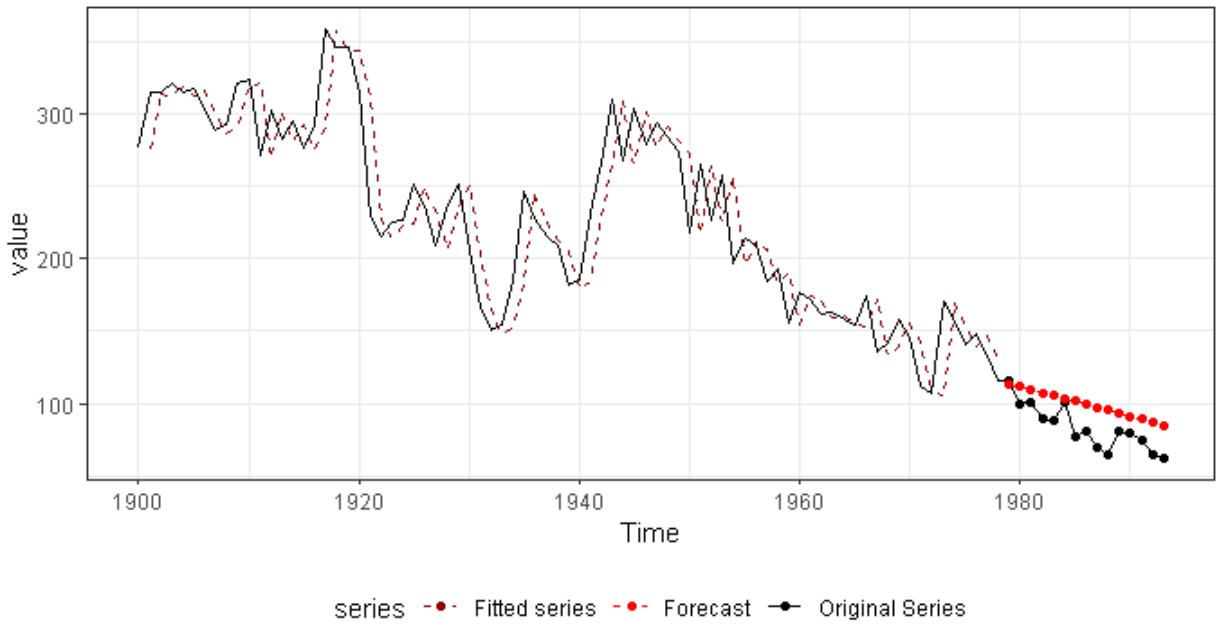

Figura 11.13: Gráfico da série Castor2 utilizando o método rwf-drift

11.6 série venda mensal dos antigos passes urbanos L12 dos transportes de Lisboa

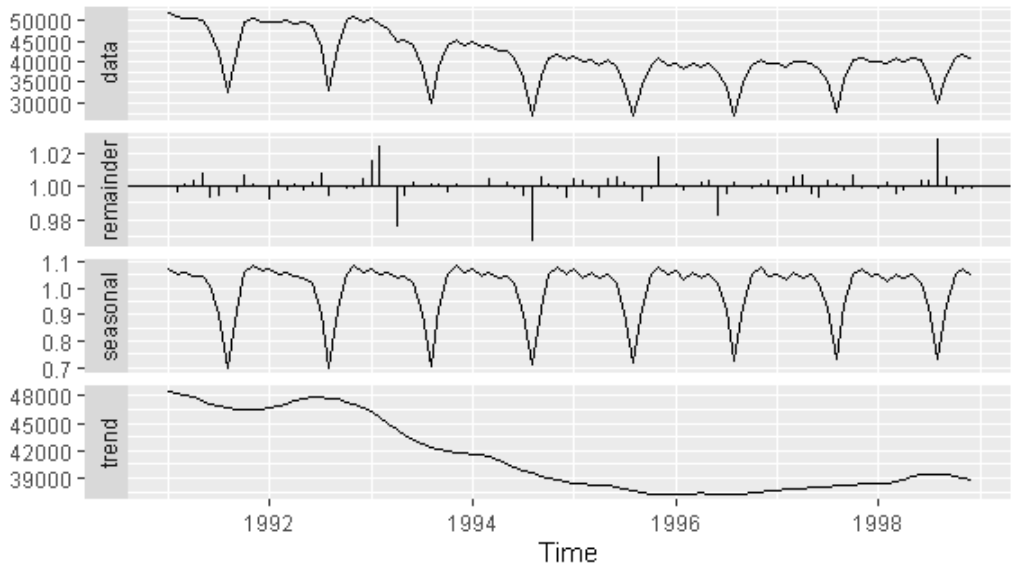

Figura 11.14: Decomposição da série passe Lisboa utilizando o método X11

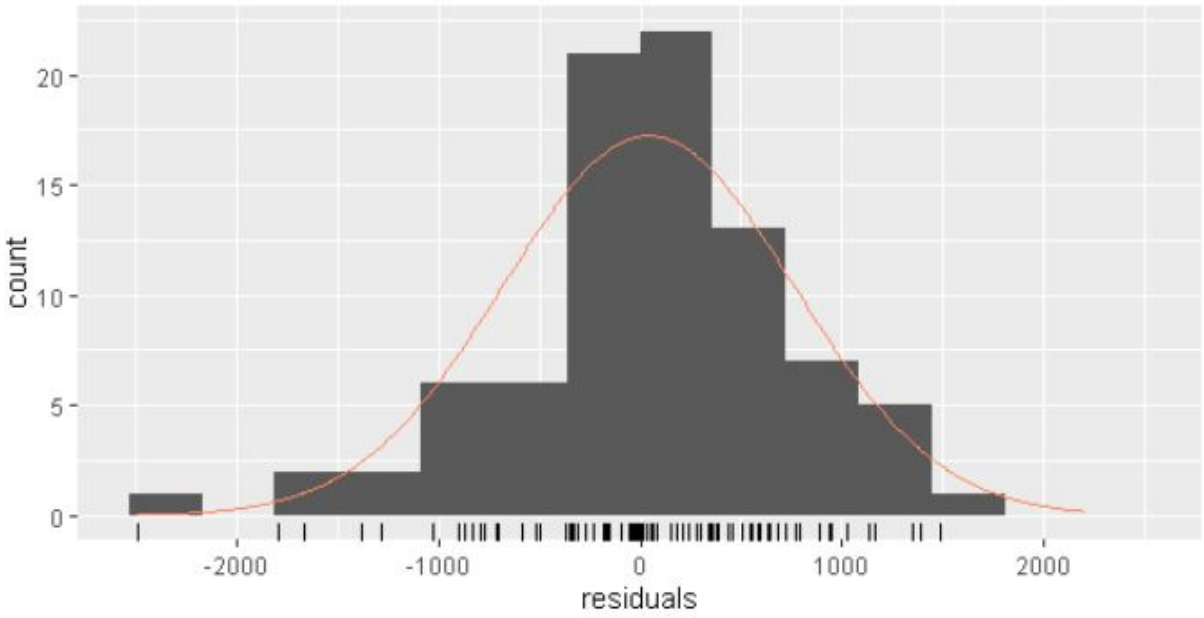

Figura 11.15: Residuos do modelo da série passe Lisboa

Quanto as séries com tendência e sazonalidade nomeadamente a série Passe Lisboa, tabela 11.6, nos ilustra que o modelo StructTS (MAPE = 67.6), e o modelo splinef (MAPE = 32.2) não são modelos mais adequado para ajustamento e previsão de dados que apresentam sazonalidade e tendência, enquanto que o modelo stlm-ets (MAPE = 3.51), o modelo nnetar (MAPE = 3.86) e o modelo auto.arima (MAPE = 3), se pode considerar que obtiveram um bom desempenho.

$\underline{model}$	$\underline{RMSE}$	$\underline{MPE}$	$\underline{MAPE}$	$\underline{MASE}$	$\underline{melhor-modelo}$	$\underline{runtime-model}$
auto.arima	1377.188	2.772366	3.004072	0.5015162	auto.arima	18.23
bats	2493.802	5.334067	5.731775	0.9627913	auto.arima	0.91
croston	3503.518	1.568445	7.939908	1.2852895	auto.arima	0.26
ets	1856.606	4.034875	4.234836	0.7033357	auto.arima	1.58
hybrid	2152.826	4.878309	4.996036	0.8370208	auto.arima	8.44
meanf	4985.321	-10.571556	10.571556	1.6408713	auto.arima	0.02
naive	3504.431	-3.472032	6.214259	0.9291711	auto.arima	0.00
nnetar	1939.342	3.808393	3.865620	0.6730238	auto.arima	1.28
rwf	3504.431	-3.472032	6.214259	0.9291711	auto.arima	0.00
rwf-drift	3851.512	2.181464	8.392793	1.3729994	auto.arima	0.00
splinef	14916.067	31.283175	32.226865	5.6319546	auto.arima	0.03
stlm-arima	3760.698	8.350666	8.565592	1.4202154	auto.arima	0.05
stlm-ets	1719.882	3.155159	3.517761	0.5574557	auto.arima	0.03
StructTS	29189.121	-67.614555	67.614555	11.5085695	auto.arima	0.04
tbats	2172.889	4.811778	5.056081	0.8451466	auto.arima	7.84
thetaf	2819.907	6.346643	6.560475	1.1035786	auto.arima	0.02
tslm	6082.552	14.952924	14.952924	2.5085152	auto.arima	0.00

Tabela 11.6: Resumo do desempenho dos diferentes modelos utilizando diferentes métricas da série passe lisboa

11.7 Série números de passageiros de linha aérea em Austrália

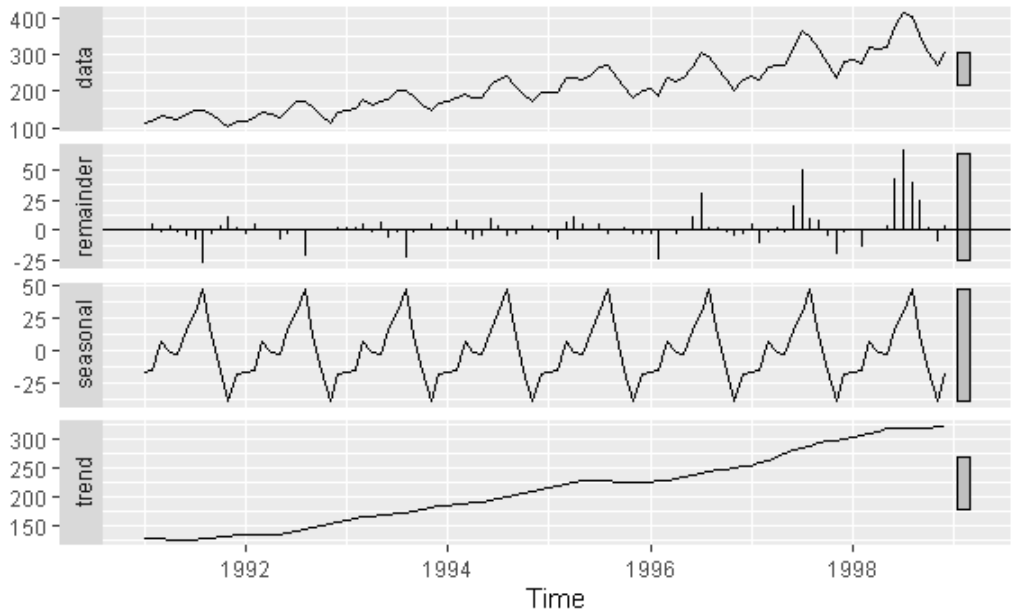

Figura 11.16: Decomposição da série passe aérea utilizando a método STL

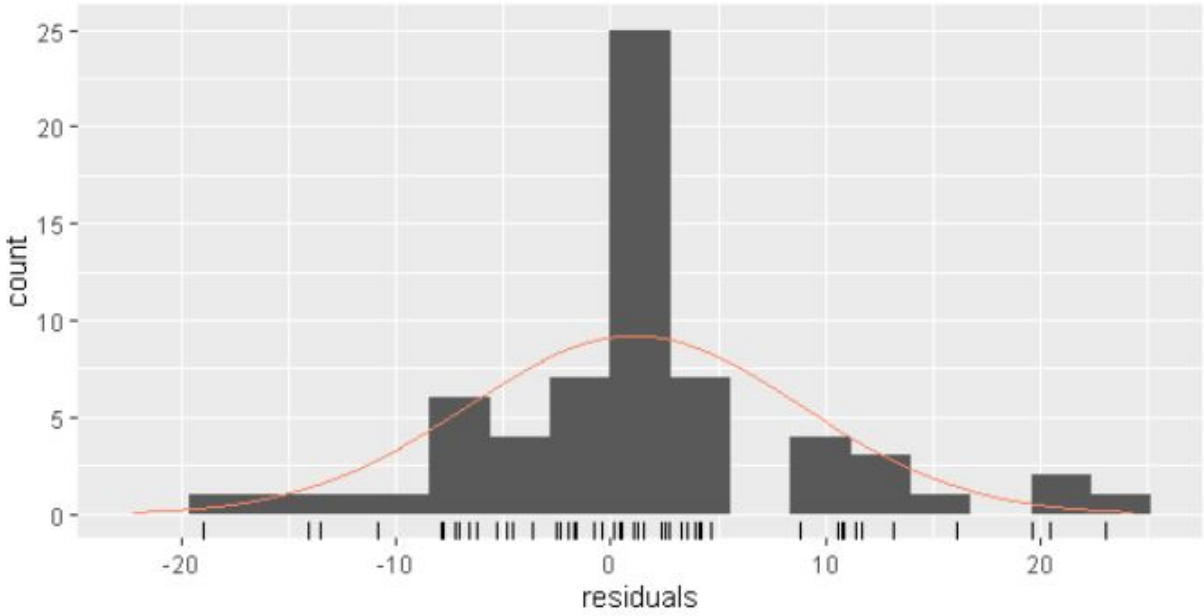

Figura 11.17: Residuos do modelo da série passe aérea

Outra série que apresenta tendência e sazonalidade, é a série passe aérea, cujo o resultado se apresenta na tabela 11.7, onde se pode considerar que os resultados não foram globalmente bons, e o modelo StructTS (MAPE = 62.76) teve o pior desempenho, demonstrando assim como na série anterior, que este não é o melhor modelo para dados com tendência e sazonalidade, sendo o melhor modelo o tbats (MAPE = 6.9).

$model$	$RMSE$	MPE	$MAPE$	$MASE$	$melhor - modelo$	$runtime - model$
auto.arima	29.77920	29.81213	9.164496	1.295241	tbats	63.64
bats	57.20026	57.20026	17.860347	2.485167	tbats	1.00
croston	73.19342	73.19342	22.066876	3.180018	tbats	0.26
ets	31.94845	33.45369	10.360069	1.453455	tbats	1.45
hybrid	34.46892	34.46892	10.603149	1.497563	tbats	4.95
meanf	123.22222	123.22222	38.805270	5.353608	tbats	0.00
naive	77.12500	77.12500	23.382283	3.350833	tbats	0.00
nnetar	51.05767	51.05767	16.167340	2.218291	tbats	0.55
rwf	77.12500	77.12500	23.382283	3.350833	tbats	0.00
rwf-drift	56.52641	57.37031	17.149037	2.492555	tbats	0.00
snaive	67.20833	67.20833	21.524036	2.919986	tbats	0.01
splinef	-145.27544	146.36433	47.983511	6.359059	tbats	0.02
stlm-arima	37.80326	38.45286	11.639888	1.670653	tbats	0.11
stlm-ets	37.79033	38.44101	11.636026	1.670138	tbats	0.01
StructTS	198.66218	198.66218	62.761413	8.631231	tbats	0.05
tbats	21.32992	22.35148	6.900199	0.971100	tbats	2.00
thetaf	41.62984	41.73951	12.935559	1.813447	tbats	0.02
tslm	26.75556	27.99653	8.262382	1.216359	tbats	0.00

Tabela 11.7: Resumo do desempenho dos diferentes modelos utilizando diferentes métricas da série passe aérea

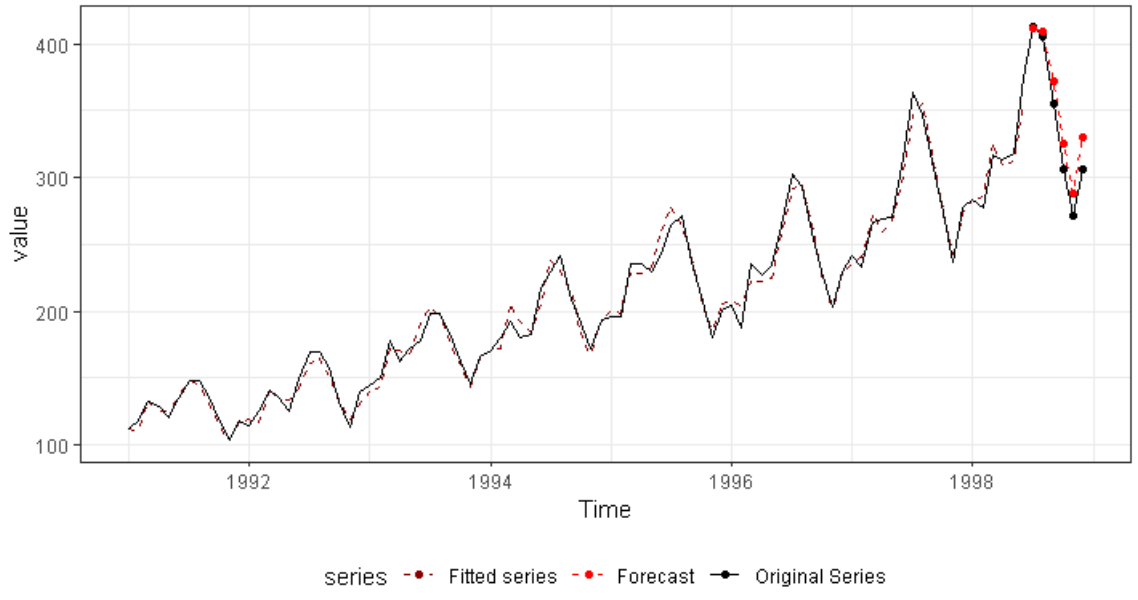

Figura 11.18: Gráfico da série passe aérea utilizando o método tbats

11.8 Série elevação média da superfície da água no Lago Huron

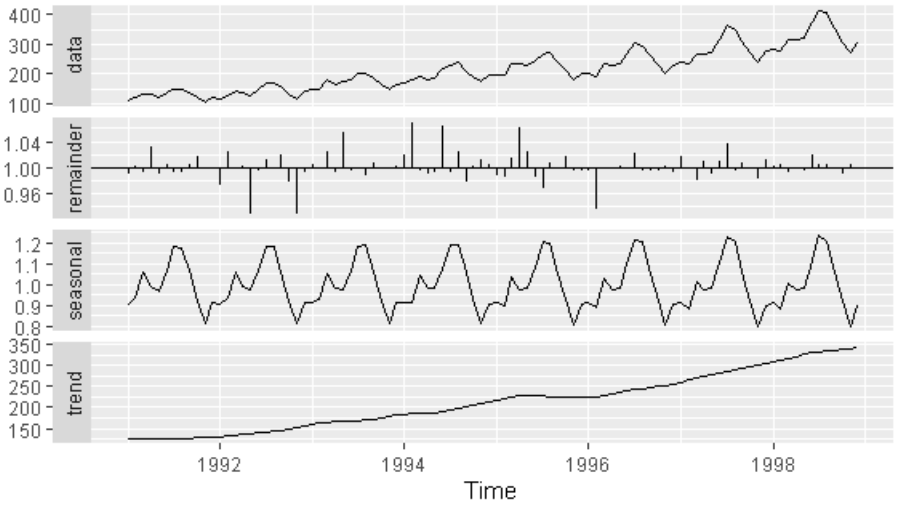

Figura 11.19: Decomposição da série Lago Huron utilizando o método X11

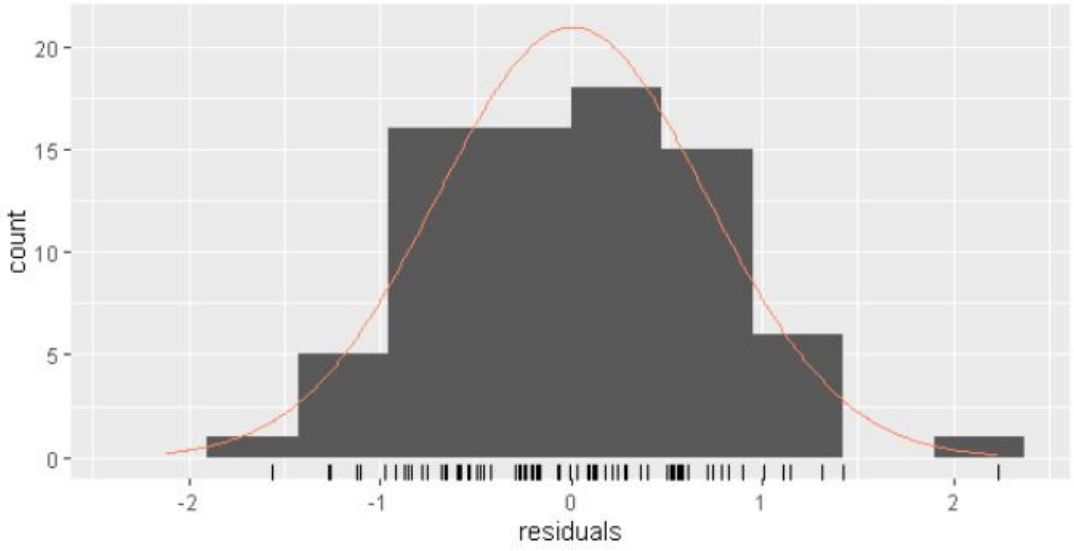

Figura 11.20: Residuos do modelo da série Lago Huron

Os resultados da série Lago Huron, tabela 11.8, nos mostra um bom e igual desempenho dos modelos naive (MAPE = 0.78) e rwf (MAPE = 0.78), enquanto que o modelo meanf (MAPE = 47.2), teve a pior perfomance.

$model$	$RMSE$	MPE	$MAPE$	$MASE$	$melhor - modelo$	$runtime - model$
auto.arima	0.03640118	-2.04091998	2.1450530	0.24614558	naive	11.28
bats	0.07011173	-4.36488806	4.3648881	0.50142210	naive	1.36
croston	0.14652360	10.89883252	10.8988325	1.25954617	naive	0.44
ets	0.02747879	-1.55784111	1.6296574	0.18702439	naive	1.79
hybrid	0.03214833	-1.82740205	1.8986256	0.21788073	naive	13.92
meanf	0.63283154	47.22133723	47.2213372	5.45461460	naive	0.00
naive	0.01103781	-0.01923947	0.7844705	0.09051724	naive	0.00
nnetar	0.04129909	3.00428536	3.0042854	0.34693584	naive	0.83
rwf	0.01103781	-0.01923947	0.7844705	0.09051724	naive	0.00
rwf-drift	0.04331444	-2.54033162	2.5745319	0.29549334	naive	0.02
snaive	0.13274160	9.87113788	9.8711379	1.14080460	naive	0.00
splinef	0.05335240	-3.21926584	3.2192658	0.36960722	naive	0.04
stlm-arima	0.05920201	-4.04521270	4.0452127	0.46555341	naive	0.10
stlm-ets	0.02447855	-1.78456867	1.7845687	0.20574977	naive	0.03
tbats	0.06923813	-4.30581696	4.3058170	0.49462602	naive	3.97
thetaf	0.02340403	-1.12757922	1.3329239	0.15287308	naive	0.00
tslm	0.18534095	13.77167227	13.7716723	1.59167934	naive	0.02

Tabela 11.8: Resumo do desempenho dos modelos utilizando diferentes métricas da série LAGO

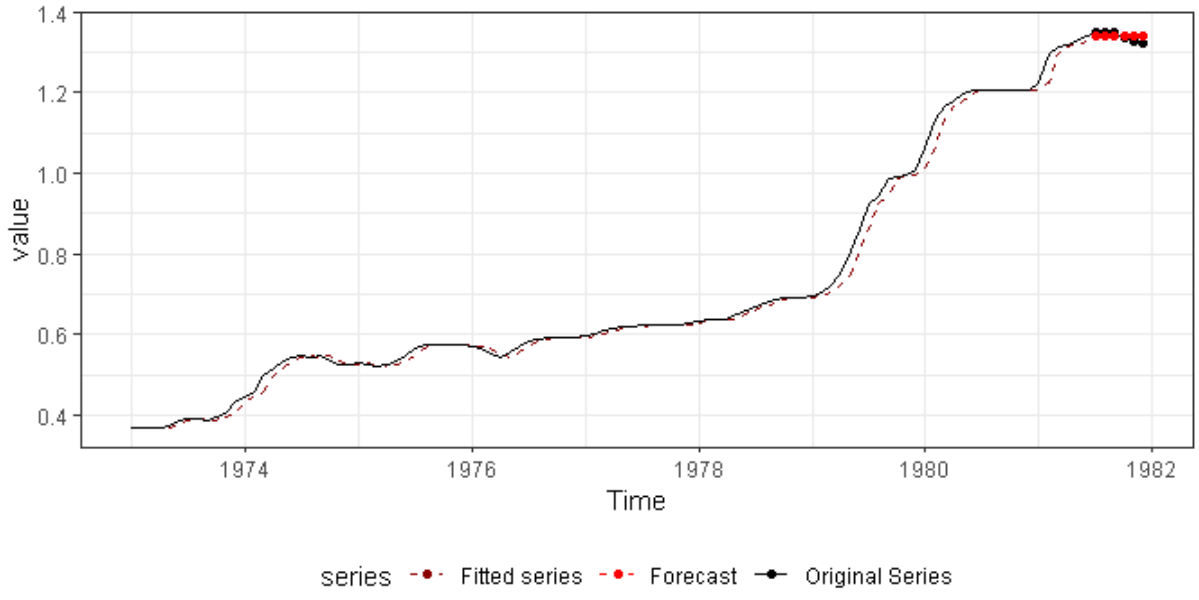

Figura 11.21: série Lago Huron

11.9 Série produção trimestral de automóveis de passageiros no Reino Unido

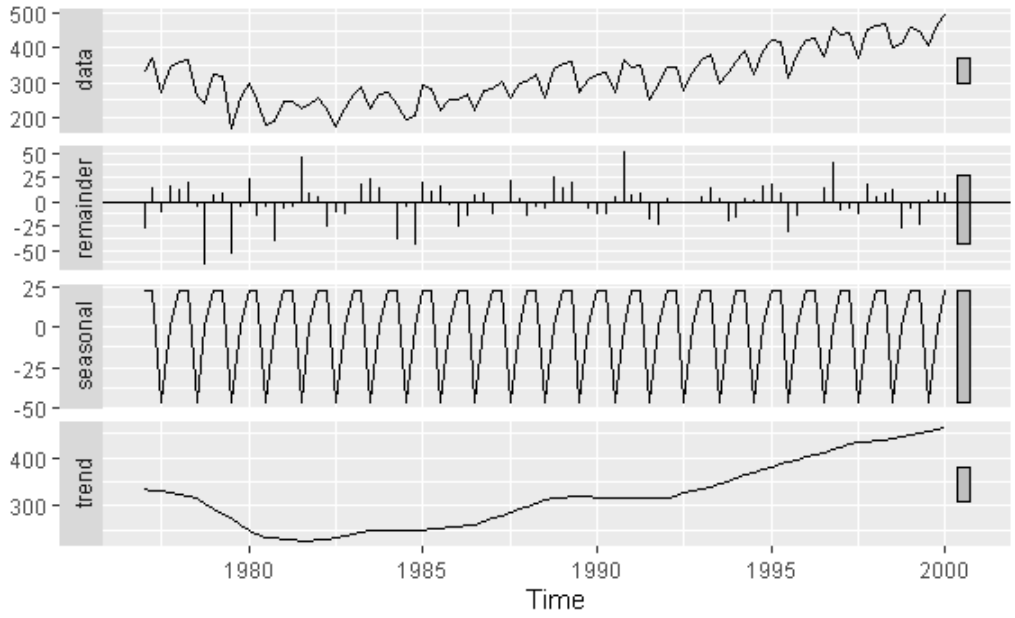

Figura 11.22: Decomposição da série produção trimestral dos carros em Reino Unido utilizando o método STL

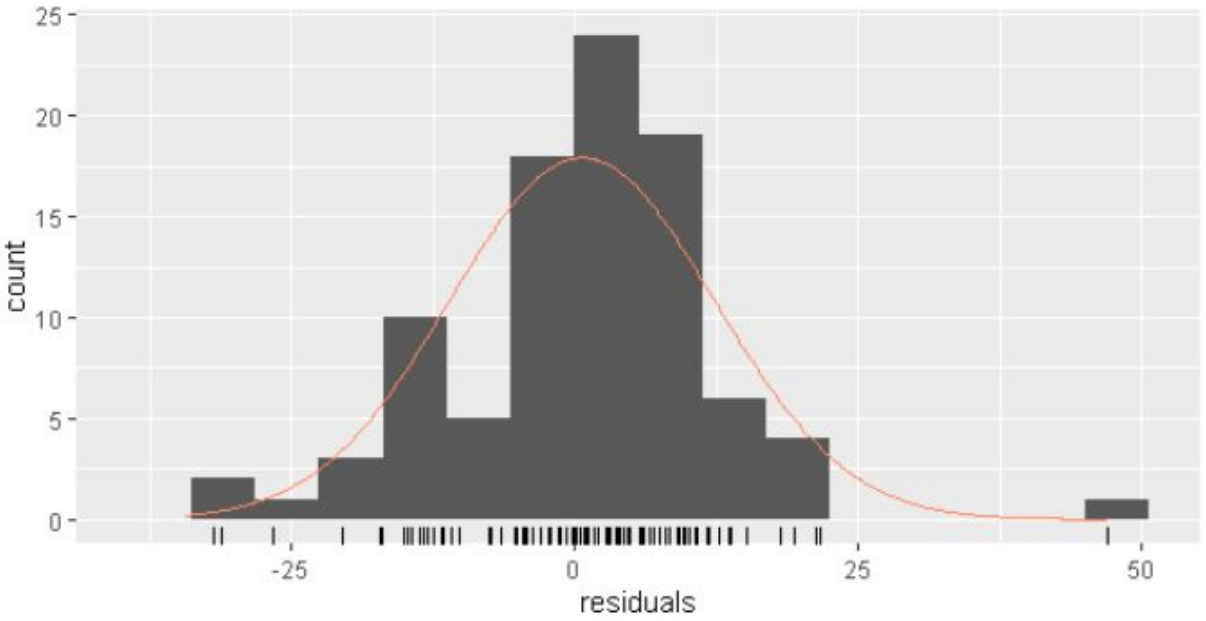

Figura 11.23: Residuos do modelo da série Carro Reino Unido

Uma outra série que foi analizada é a série produção de carros no Reino Unido, onde os resultados se apresentam na tabela 11.9. Para essa série o melhor modelo encontrado foi o modelo tbats (MAPE = 3.58), com vários outros modelos tendo um desempenho próximo deste, como o modelo bats (MAPE = 3.75)], ou o modelo stlm-arima (MAPE = 3.65). Os modelos naive (MAPE = 10.05) e meanf (MAPE = 20.62), tiveram os piores desempenhos no ajustamento e previsão.

$\underline{model}$	$\underline{RMSE}$	$\underline{MPE}$	$\underline{MAPE}$	$\underline{MASE}$	$\underline{melhor-modelo}$	$\underline{runtime-model}$
auto.arima	25.77037	3.7898451	5.195787	0.6549613	tbats	8.83
bats	19.32905	-0.6830550	3.748072	0.4675080	tbats	1.05
croston	21.60674	0.7560443	4.595173	0.5954876	tbats	0.51
ets	25.01912	-0.5672626	5.358748	0.6686833	tbats	0.48
hybrid	20.53256	1.8753628	4.066330	0.5081112	tbats	3.06
meanf	87.93646	20.6168887	20.616889	2.6688146	tbats	0.00
naive	45.26785	-10.0534045	10.053404	1.2507796	tbats	0.00
nnetar	30.55617	-4.1310957	6.168039	0.7844292	tbats	1.00
rwf	45.26785	-10.0534045	10.053404	1.2507796	tbats	0.00
rwf-drift	52.04998	-11.9416131	11.941613	1.4934463	tbats	0.01
snaive	37.66226	5.2248557	7.649343	0.9742706	tbats	0.00
splinef	49.69381	9.9934873	9.993487	1.3150928	tbats	0.07
stlm-arima	18.52361	0.4061228	3.645441	0.4543839	tbats	0.23
stlm-ets	18.40248	-1.3408300	3.885603	0.4878846	tbats	0.03
StructTS	19.15528	0.3766622	4.137786	0.5194408	tbats	0.02
tbats	17.48323	0.7144799	3.584812	0.4509977	tbats	6.91
thetaf	25.56135	-1.0252457	5.615690	0.7062040	tbats	0.00
tslm	31.61759	-6.2143748	6.997562	0.8969000	tbats	0.00

Tabela 11.9: Resumo do desempenho dos diferentes modelos utilizando diferentes métricas da série carro no Reino Unido

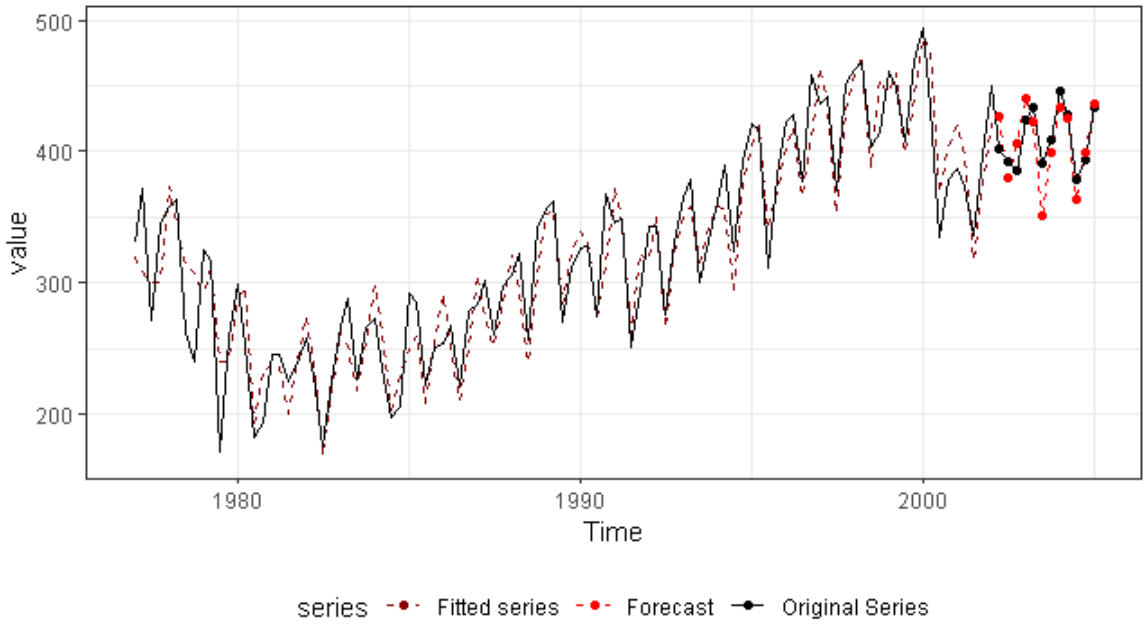

Figura 11.24: Modelo tbats da série passe aerea