AVALIAÇÃO EM LARGA ESCALA DE MODELOS CLÁSSICOS E MODERNOS PARA PREVISÃO DE DEMANDA EM COMÉRCIO ELETRÔNICO

Diego Tertuliano da Silva

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Daniel Ratton Figueiredo

Rio de Janeiro
Outubro de 2020

AVALIAÇÃO EM LARGA ESCALA DE MODELOS CLÁSSICOS E MODERNOS PARA PREVISÃO DE DEMANDA EM COMÉRCIO ELETRÔNICO

Diego Tertuliano da Silva

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Daniel Ratton Figueiredo

Aprovada por: Prof. Daniel Ratton Figueiredo
Prof. Daniel Sadoc Menasche
Prof. Felipe Maia Galvão França

RIO DE JANEIRO, RJ – BRASIL
OUTUBRO DE 2020

Silva, Diego Tertuliano da

Avaliação Em Larga Escala de Modelos Clássicos e Modernos para Previsão de Demanda em Comércio Eletrônico/Diego Tertuliano da Silva. – Rio de Janeiro: UFRJ/COPPE, 2020.

XI, 77 p.: il.; 29, 7cm.

Orientador: Daniel Ratton Figueiredo

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2020.

Referências Bibliográficas: p. 69 – 72.

1. previsão de demanda. 2. *big data*. 3. redes neurais. I. Figueiredo, Daniel Ratton. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Agradecimentos

Agradeço aos meus pais, Aroldo e Ruth, por acreditarem em meu potencial, incentivarem e por me apoiarem em todos os momentos. Meus pais e meu irmão Diogo são os exemplos que me fazem tentar dar o meu melhor todos os dias. Sem eles essa jornada nunca teria sido possível.

Agradeço ao meu orientador, Prof. Daniel Ratton Figueiredo, pela amizade (com muitos cafés e trilhas); pela compreensão nas mudanças de tema; pelos ensinamentos nos mais diversos assuntos; pela ética e rigor acadêmico. A oportunidade de ter sido orientado por ele foi uma experiência enriquecedora que levarei para o restante da vida.

Agradeço a todos os amigos que fizeram parte deste processo nestes três anos de mestrado e os anteriores que me acompanham desde a graduação. Pela amizade e também pela curiosidade intelectual, sempre dispostos a debater sobre qualquer tema. Em especial: Alexandre Moreira, Bernardo Killer, Danielle Castelo Branco, Diego Ximenes, Erick Pires, Gabriela Lewenfus, Iago Leal, Laura Moraes, Pedro Aragão, Raul Gabrich, Rodrigo Peres, Rodrigo Zhou e Vinícius Layter Xavier.

Agradeço ao Prof. Daniel Sadoc Menasche e o Prof. Felipe Maia Galvão França pela oportunidade de tê-los como avaliadores deste trabalho.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

AVALIAÇÃO EM LARGA ESCALA DE MODELOS CLÁSSICOS E MODERNOS PARA PREVISÃO DE DEMANDA EM COMÉRCIO ELETRÔNICO

Diego Tertuliano da Silva

Outubro/2020

Orientador: Daniel Ratton Figueiredo

Programa: Engenharia de Sistemas e Computação

Previsões realizadas para o planejamento de demanda futura são cruciais para se ter um inventário disponível no lugar certo e no momento certo. Neste trabalho a previsão de demanda em larga escala, fruto de uma colaboração com a empresa de plataforma de comércio unificada VTEX, foi abordada através de técnicas clássicas e modernas de previsão de demanda na literatura. A partir de dados reais, séries temporais em diferentes escalas de tempo (dia, semana e mês) e diferentes segmentos (produto, categoria e conta) foram construídas e analisadas. O grande volume de dados trouxe diversos desafios computacionais: desde o tratamento dos dados brutos para geração das séries temporais, sendo necessária a utilização do motor de processamento distribuído Apache Spark; até o treinamento dos parâmetros dos modelos. Neste trabalho um *pipeline* para o *ETL* dos dados brutos foi desenvolvido na plataforma de serviços de computação em nuvem (AWS) utilizando tecnologias para processamento de dados em larga escala, assim como a criação de um *framework* para a comparação dos modelos. A caracterização das séries temporais mostrou uma grande diversidade mesmo em séries de uma mesma escala de tempo e segmento, com comportamento de cauda pesada em algumas estatísticas. Mostramos que a utilização de diferentes modelos para cada série temporal obteve melhores desempenhos, devido as diferentes características entre as séries. Se nos restringirmos a um único modelo para todas as séries temporais, o modelo baseado em uma rede neural probabilística recorrente e autoregressiva (DeepAR) obteve o melhor desempenho geral. A estratégia utilizando uma combinação de modelos obteve ganhos MAPE de até 80.6% sobre o melhor modelo no segmento de contas e escala de tempo mensal.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

LARGE SCALE EVALUATION OF CLASSIC AND MODERN DEMAND FORECASTING MODELS FOR E-COMMERCE

Diego Tertuliano da Silva

October/2020

Advisor: Daniel Ratton Figueiredo

Department: Systems Engineering and Computer Science

Forecasting demand is crucial for having the right inventory available at the right time at the right place. In this work a large scale demand forecasting problem, result of an industry collaboration with the unified commerce platform VTEX, was approached through classical and modern techniques of demand forecasting in the literature. From real data, time series in different timescales (day, week, and month) and different segments (product, category, and account) were built and analyzed. The large volume of data brought several computational challenges: from processing the raw data to build the time series, requiring a distributed data processing engine, Apache Spark; to training the model parameters. A pipeline for the ETL of raw data was developed on the cloud computing services platform (AWS) using technologies for large-scale data processing, as well as developing a framework for the comparison of models. The characterization of time series exhibited a great diversity even in series of the same timescale and segment, with heavy tail behavior in some statistics. We show that the use of different models for each time series achieved better performances, due to the different characteristics between the series. If we restrict ourselves to a single model for all time series, the model based on a recurrent and autoregressive probabilistic neural network (DeepAR) achieved the best overall performance. The strategy using a combination of models achieved MAPE gains up to 80.6% over the best model in the segment of accounts and monthly timescale.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Forecast. Verb: Predict or estimate (a future event or trend).
Noun: A calculation or estimate of future events, especially coming weather or a financial trend. Oxford English Dictionary.

Previsão (*forecast*), como definido pelo *Oxford English Dictionary*, é o ato de prever, estimar ou calcular um evento futuro, ou tendência - como o tempo ou tendências do mercado. A previsão destas tendências e eventos futuros, é uma tarefa comum em negócios, uma vez que ajuda na tomada de decisão de diferentes domínios, como produção e transportes de materiais [5], e provê uma maneira de guiar decisões estratégicas de longo prazo [4]. O objetivo de *forecasting* é prever o futuro da maneira mais acurada possível, dadas todas as informações disponíveis, incluindo dados históricos e o conhecimento de qualquer outro evento que possa influenciar o futuro [6]. Produzir previsões de grandes coleções de itens, como por exemplo a quantidade de chuva em milímetros para todas as cidades do mundo, é uma tarefa extremamente relevante e igualmente desafiadora. Modelos clássicos falham em capturar padrões complexos nos dados, enquanto técnicas multivariáveis possuem dificuldade em escalar para problemas com muitos itens [7].

Previsão tem um papel fundamental na automatização e otimização de processos operacionais na maioria dos negócios, permitindo que a tomada de decisão seja baseada em dados passados [8]. No varejo - foco deste trabalho - decisões de quais produtos estocar, quando realizar pedidos e onde armazená-los dependem de previsões da demanda futura de diferentes regiões. Por exemplo, em computação na nuvem, a estimativa futura de uso de serviços e infraestrutura guiam o planejamento da capacidade computacional. Previsões de consumo de energia são usadas para planejar e otimizar a geração de energia, e agendamento de trabalhadores em armazéns e fábricas dependem do cálculo da carga de trabalho futura [9].

Séries temporais, chamadas desta maneira por ocorrerem em uma sequência de eventos no tempo, são os objetos primários deste trabalho e definidos formalmente

no Capítulo 2. Um exemplo de série temporal é a quantidade de chuva de uma cidade por dia, desde o início da coleta desses dados. Os métodos prevalentes de previsão em estatística e econometria são desenvolvidos para o contexto de previsões de séries temporais individuais ou de pequenos grupos, necessitando de engenharia de atributos e criação de modelos por especialistas da área de domínio [9]. Recentemente, ocorreu uma mudança destes métodos clássicos para métodos totalmente automatizados baseados estritamente em dados. Esta quebra de paradigma pode ser atribuída à disponibilidade de grandes conjuntos de dados, sendo agora possível aprender diretamente dos dados sem uma quantidade significativa de trabalho manual [10]. Este trabalho realiza a comparação entre os métodos clássicos e os novos métodos e baseados estritamente em dados.

Este trabalho de dissertação foi realizado em colaboração com a empresa VTEX, onde o autor trabalha como cientista de dados no problema de previsão de demanda. A VTEX [11] é uma multinacional brasileira de tecnologia com foco em *cloud commerce* desenvolvedora da plataforma VTEX *Cloud commerce*, disponibilizada no mercado como SaaS, com atuação global e com clientes como Walmart, Whirlpool, Lego, Disney, L'oreal, Sony, Coca-Cola, Staples, O Boticário, Nespresso, Ambev, Tramontina, Bosch e outras 4.500+ lojas em mais de 28 países [12]. Consequentemente, a VTEX gera diariamente uma enorme quantidade de dados, inclusive sobre vendas dos produtos de seus clientes, que será o foco deste trabalho. Todos os dados utilizados neste trabalho são reais e por questões de privacidade dos clientes foram anonimizados.

1.1 Motivação

Foi notado que em diversos momentos produtos que são responsáveis por grande parte do faturamento dos clientes não está disponível no estoque e não podem ser vendidos. Podemos observar este comportamento na Figura 1.1, no qual praticamente toda a concentração das vendas ocorrem para os três SKUs (*stock-keeping unit*, um exemplo de SKU é o modelo de um tênis de uma determinada marca de tamanho 40) mais vendidos. Desta forma, é de fundamental importância o planejamento de inventário de forma a evitar estas situações, aumentando o faturamento dos clientes VTEX.

O planejamento de inventário apresenta diversos pontos de melhora para um varejista: a otimização do estoque de acordo com altas e baixas na demanda; diminuição dos custos de estoque nos armazéns; a criação de estratégias de propaganda baseada em dados e maior agilidade às mudanças de demanda do mercado [13]. Desta forma, a previsão de demanda é um aspecto importante do planejamento de inventário.

Figura 1.1: SKUs mais vendidos para um varejista, este comportamento é semelhante para outros clientes VTEX

1.2 Descrição do problema

Um bom planejamento do inventário requer uma previsão acurada do futuro: uma estimativa errada para cima pode criar diversos produtos encalhados (além do custo da compra desses produtos, implica em gastos de armazenagem); uma estimativa errada para baixo representa uma perda de oportunidade de vendas. Desta forma é vital o planejamento adequado do estoque de acordo com as características do negócio. Previsões realizadas para o planejamento são cruciais para se ter um inventário disponível no lugar certo e no tempo certo.

A VTEX possui diferentes segmentos de clientes, tais como moda, eletroeletrônicos e hipermercados. Isto significa diferentes sazonalidades e causas que afetam de maneiras diferentes cada uma das demandas. Devido a essas variadas características, diferentes modelos podem ser criados de acordo com os agrupamentos, de forma a maximizar a extração de informação global dos padrões dos dados. Entretanto, há muitas formas de agregar a informação, seja na escala temporal (dia, semana, etc) ou na escala de vendas (produto, categoria, loja). Intuitivamente, a agregação dos dados possui papel central na previsão, tanto na acurácia quanto na utilidade.

1.3 Contribuição

O objetivo deste trabalho é avaliar e comparar diferentes modelos de previsão, tanto clássicos quanto recentes, quando aplicados ao problema de previsão de demanda. Em particular, o trabalho possui as seguintes contribuições:

- Por conta da colaboração com a indústria, permitiu acesso a infraestrutura computacional sob demanda para processamento de larga escala que possibilitaram os experimentos realizados, através da AWS. Esta infraestrutura foi utilizada desde o processamento dos dados brutos que totalizam aproximadamente 14 terabytes, sendo assim necessário a utilização de motores de computação distribuída, neste trabalho *jobs* em Spark [14] rodando com o serviço de Elastic MapReduce (EMR) utilizando 408 vCPUs e 864 GiB, até a computação paralela de modelos de previsão em máquinas com alto números de CPUs.

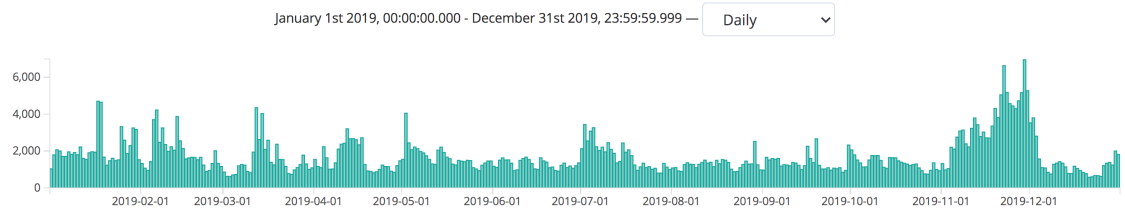
- Uso de dados reais de larga escala, agregados por tempo (dia, semana e mês) e por venda (produto, categoria e volume total por conta). A dificuldade combinatória para a escolha do nível de segregação e intervalo de tempo, aliado aos diferentes comportamentos inerentes ao conjunto de dados utilizados trouxe uma gama de desafios que normalmente não são encontrados em conjuntos de dados sintéticos ou *baseline* clássicos.

- Comparação de modelos clássicos e modernos quando aplicados a séries temporais para prever demanda, com a utilização de previsões probabilísticas no caso moderno de forma a possibilitar a quantificação de risco de previsões;

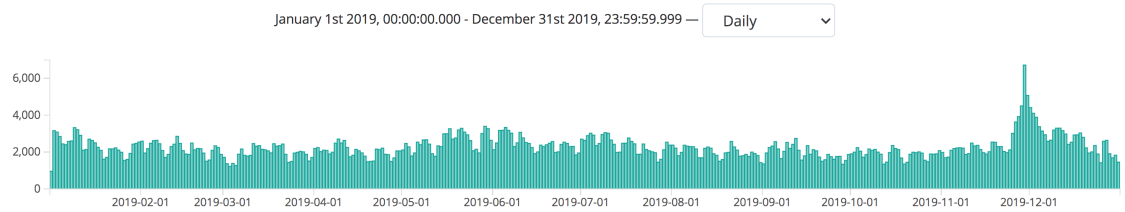
Além disso, pelo conhecimento do autor, até hoje nenhum estudo realizou a comparação de modelos de previsão (clássicos ou não) com uma diversidade de séries temporais semelhante a este trabalho (conjuntos de dados relacionados a uma única empresa existem, ex.: [15]). No conjunto de dados deste trabalho, a influência de fatores externos é diferente em diferente séries temporais, pois a venda de produtos é influenciada pelos mais diversos motivos, por exemplo, um feriado como dia das mães aumenta de maneira não uniforme as vendas de um varejista em sua totalidade, enquanto o oposto acontece com outro varejista. Podemos observar este comportamento na Figura 1.2 através dos pedidos de duas lojas no período de 2019. A mesma situação representada de forma global ocorre dentro de categorias de produtos de um varejista, ou seja, categorias são afetadas de maneira diferente de acordo com um feriado (por exemplo comidas típicas de um feriado em um supermercado). Do ponto de visto prático, diversos problemas surgem na aplicação dos modelos quando

o número de itens é muito grande, de forma que métodos não paralelizáveis e métodos computacionalmente intensivos se tornam inviáveis devido ao longo tempo de execução.

Desta maneira, através deste trabalho, podemos ter uma noção mais realista da aplicação de modelos de previsão de séries temporais na previsão de demanda baseada em dados reais, tanto em relação à sua acurácia quanto aos desafios computacionais dos modelos.



(a) Demanda da loja A no ano de 2019



(b) Demanda da loja B no ano de 2019

Figura 1.2: Influência de eventos na demanda de séries temporais

1.4 Tecnologias utilizadas

Este trabalho foi desenvolvido utilizando Python [16] como linguagem de programação; Spark [14] para a transformação dos dados brutos em séries temporais; StatsModels [17] e GluonTS [18] para os modelos de previsão das séries temporais, assim como a parametrização dos modelos a partir dos dados reais.

1.5 Organização

O restante deste texto está organizado da seguinte forma: o Capítulo 2 descreve séries temporais e métodos clássicos de previsão como exposto em [4]. O Capítulo 3 apresenta aspectos teóricos sobre redes neurais, em especial a seção 3.4 descreve redes neurais recorrentes como exposto em [3] e a seção 3.5 descreve o modelo DeepAR como exposto em [19]. O Capítulo 4 explica como podemos mensurar os erros de um modelo e garantir que a acurácia encontrada em tempo de treinamento continue fora da amostra. O Capítulo 5 explica a construção e caracterização do conjunto de dados analisado neste trabalho, incluindo os diferentes níveis de agregação. O Capítulo 6

avalia os diferentes modelos usando o conjunto de dados proposto. Finalmente, o Capítulo 7 apresenta uma breve conclusão e possíveis trabalhos futuros.

Capítulo 2

Modelos clássicos de previsão

Este capítulo apresenta rapidamente alguns dos principais modelos clássicos de previsão, fazendo uma revisão de seus parâmetros e uso. O problema de previsão também é enunciado e ilustrado no contexto dessa dissertação, na aplicação de previsão de demanda.

Nos estágios iniciais de um projeto de previsão de demanda, diversas decisões precisam ser tomadas. Por exemplo, em um cenário de e-commerce é necessário realizar as seguintes perguntas:

1. É necessário uma previsão para cada produto, para um grupo de produtos ou por categorias?
2. Previsões por lojas separadas ou um agregado de lojas?
3. Qual é a escala de tempo para a previsão? hora, dia, mês, ano?
4. Horizonte da previsão: quanto tempo no futuro se deseja fazer a previsão?
5. Quão frequente as previsões precisam ser realizadas?

Essas perguntas serão consideradas no decorrer dessa dissertação e principalmente na avaliação experimental dos modelos de previsão.

Uma série temporal é qualquer fenômeno observado sequencialmente durante o tempo. De maneira geral os intervalos entre os pontos de observação são regulares: por minuto, hora, dia, mês, etc.

Exemplos de séries temporais são:

- Uma ação na bolsa de valores, por exemplo: PETR4, VALE3;
- Quantidade de chuva por dia ou mês;
- Lucro anual de uma empresa

Figura 2.1: Gráfico da ação PETR4 (Petrobras) observado diariamente

A série temporal dia a dia da ação PETR4 pode ser observada na Figura 2.1, no período ente 19/12/2018 até 12/12/2019.

Existem dois tipos de previsões para séries temporais: quando não existem dados disponíveis, métodos qualitativos precisam ser usados. Neste trabalho utilizamos somente métodos quantitativos, no qual informação do passado está disponível.

O objetivo de previsão de séries temporais é estimar como a sequência de observações continuará no futuro. Os métodos mais simples de previsão utilizam somente informações das variáveis a serem estimadas, sem identificar os fatores que influenciam esse comportamento. Métodos mais sofisticados podem utilizar informações externas como indicadores econômicos, informações de produtos concorrentes, entre outros.

2.1 Previsão de séries temporais e variáveis preditivas

Suponha que queiramos prever a demanda de eletricidade para o verão, um modelo com variáveis preditivas poderia ser da seguinte forma:

$$Demanda = f(\text{temperatura atual}, \text{indicadores econômicos}, \text{população}, \text{horário do dia}, \text{dia da semana}, \text{erro}) \tag{2.1}$$

A igualdade entre a demanda e a função f vem do fato de incorporarmos o termo de erro, já que flutuações aleatórias e efeitos de outras variáveis relevantes podem não estar incluídas no modelo.

Analisando o mesmo problema como uma previsão de série temporal, a função de demanda torna-se

$$Demanda_{t+1} = f(Demanda_t, Demanda_{t-1}, Demanda_{t-2}, \ldots, \text{erro}) \tag{2.2}$$

Modelos que combinam os termos da Equação (2.1) com os termos da Equação (2.2) são chamados de modelos mistos, tais como modelos de regressão dinâmica, modelos de função de transferência, entre outros.

2.2 Métodos simples de previsão

Existem alguns modelos simples que geralmente são usados como pontos de referência para modelos mais complexos, embora existam casos que estes métodos são surpreendemente efetivos.

2.2.1 Método ingênuo

Uma maneira muito simples de realizar uma previsão é repetindo o valor da ultima observação, isto é:

$$\hat{y}_{T+h|T} = y_T.$$

A notação $\hat{y}_{T+h|T}$ é a estimativa de y_{T+h} baseado nas observações $y_1, \ldots, y_T$.

2.2.2 Método sazonal ingênuo

Podemos modificar o método ingênuo para séries com alta sazonalidade. Neste caso, cada previsão é igual ao último ponto observado da última temporada (ex.: mesma semana do último trimestre ou mesmo mês do último ano).

$$\hat{y}_{T+h|T} = y_{T+h-m(k+1)},$$

no qual m é o período da sazonalidade e k a parte inteira de $\frac{h-1}{m}$, ou seja $k = \lfloor \frac{h-1}{m} \rfloor$. Por exemplo $\hat{y}_{10+5|10} = y_{10+5-4(1+1)} = y_7$. Para quando a sazonalidade da série temporal é de $m = 4$ e estamos prevendo o quinto ponto a frente a partir do tempo atual $T = 10$.

2.2.3 Método da média

Seja $y_1, \ldots, y_T$ as observações passadas, então podemos denotar a previsão como sendo a média do passado, dado por

$$\hat{y}_{T+h|T} = \bar{y} = \frac{y_1 + \cdots + y_T}{T}.$$

2.2.4 Método de *drift*

Uma variação do método ingenuo é permitir que o valor da previsão aumente ou diminua ao longo do tempo no futuro, no qual a proporção de mudança (chamado de *drift*) é a alteração média dos dados históricos:

$$\hat{y}_{T+h|T} = y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1}) = y_T + h\left(\frac{y_T - y_1}{T-1}\right).$$

Isto é equivalente a criar uma reta entre a primeira e última observação e extrapolar as previsões para o futuro até $T + h$.

2.3 Suavização Exponencial

Suavização Exponencial foi proposto por [20], [21] e [22]. As previsões são realizadas com uma média ponderada das observações passadas, no qual cada um dos pesos associados a observação decai exponencialmente de acordo com o tempo.

2.3.1 Suavização Exponencial Simples (SES: Simple Exponential Smoothing)

O método mais simples de suavização exponencial é chamado de suavização exponencial simples. Esse método é adequado para previsão de dados sem tendência clara ou algum padrão sazonal. A suavização exponencial pode ser vista como uma alternativa entre os dois extremos entre o método ingênuo e o método da média. Através da suavização exponencial podemos dar maior importância para observações mais recentes do que observações de um passado distante. O método ingênuo atribui 100% de importância para a última observação, enquanto o método da média atribui igual importância para todas as observações passadas. As previsões são calculadas usando uma média ponderada, no qual o pesos decaem exponencialmente de acordo com a distância do passado, desta forma os maiores pesos são associados as observações mais recentes, formalmente:

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots, \tag{2.3}$$

onde $0 \leq \alpha \leq 1$ é um parâmetro de suavização e a previsão para o tempo $T+1$ é a média ponderada de todas as observações $y_1, \ldots, y_T$. A taxa no qual os pesos decaem é controlada pelo parâmetro α.

Para α entre 0 e 1, os pesos associados com as observações decaem exponencialmente quanto mais antigas as observações são, motivo do nome "suavização exponencial". Se α é próximo de 0, mais peso é dado para observações antigas. Se α é próximo de 1, mais peso é dado para observações recentes. Para o extremo em que $\alpha = 1$, $\hat{y}_{T+1|T} = y_T$ e a previsão se torna equivalente ao caso ingênuo.

Podemos chegar na Equação 2.3 de duas maneiras: a primeira é através da média ponderada; a segunda por uma representação em componentes que ajudarão nas próximas seções a adicionarmos novos termos a Equação de suavização exponencial.

Forma de peso ponderado

A previsão no tempo $T+1$ é igual a média ponderada entre a observação mais recente y_T e a previsão anterior $\hat{y}_{T|T-1}$:

$$\hat{y}_{T+1|t} = \alpha y_T + (1-\alpha)\hat{y}_{T|T-1}.$$

Similarmente, para o tempo $t = 1, \ldots, T$:

$$\hat{y}_{t+1|t} = \alpha y_t + (1-\alpha)\hat{y}_{t|t-1}.$$

Como esta é uma equação recursiva, precisamos de um valor inicial, neste caso ℓ_0 (necessário estimar). Logo

$$\begin{aligned}
\hat{y}_{2|1} &= \alpha y_1 + (1-\alpha)\ell_0 \\
\hat{y}_{3|2} &= \alpha y_2 + (1-\alpha)\hat{y}_{2|1} \\
\hat{y}_{4|3} &= \alpha y_3 + (1-\alpha)\hat{y}_{3|2} \\
&\vdots \\
\hat{y}_{T|T-1} &= \alpha y_{T-1} + (1-\alpha)\hat{y}_{T-1|T-2} \\
\hat{y}_{T+1|T} &= \alpha y_T + (1-\alpha)\hat{y}_{T|T-1}.
\end{aligned}$$

Realizando as devidas substituições:

$$
\begin{aligned}
\hat{y}_{3|2} &= \alpha y_2 + (1-\alpha)\left[\alpha y_1 + (1-\alpha)\ell_0\right] \\
&= \alpha y_2 + \alpha(1-\alpha) y_1 + (1-\alpha)^2 \ell_0 \\
\hat{y}_{4|3} &= \alpha y_3 + (1-\alpha)[\alpha y_2 + \alpha(1-\alpha) y_1 + (1-\alpha)^2 \ell_0] \\
&= \alpha y_3 + \alpha(1-\alpha) y_2 + \alpha(1-\alpha)^2 y_1 + (1-\alpha)^3 \ell_0 \\
&\vdots \\
\hat{y}_{T+1|T} &= \sum_{j=0}^{T-1} \alpha(1-\alpha)^j y_{T-j} + (1-\alpha)^T \ell_0.
\end{aligned}
$$

Para valores grandes de T, o último termo pode ser desprezado, levando esta forma de média ponderada até a Equação 2.3.

Forma em componentes

Uma forma alternativa de representação é a forma em componentes. Para a suavização exponencial simples, o único componente é o de nível, ℓ_t. Outros métodos podem incluir um de inclinação b_t e de sazonalidade s_t. Representações em componentes de métodos de suavização exponencial possuem uma equação de previsão e uma equação de suavização para cada componente incluído no método. A forma em componente para suavização exponencial simples é dada por

$$
\begin{aligned}
\text{Equação de previsão} \qquad & \hat{y}_{t+h|t} = \ell_t \\
\text{Equação de nível} \qquad & \ell_t = \alpha y_t + (1-\alpha)\ell_{t-1},
\end{aligned}
$$

no qual ℓ_t é o nível (ou valor suavizado) da série temporal no tempo t. A equação de previsão mostra que o valor de previsão no tempo $t+1$ é o valor estimado no tempo t. A equação de suavização para o nível (comumente abreviada de equação de nível) é o valor estimado do nível da série temporal em cada período t.

Se substituirmos ℓ_t por $\hat{y}_{t+1|t}$ e ℓ_{t-1} por $\hat{y}_{t|t-1}$ na equação de nível, recuperamos a forma de média ponderada da suavização exponencial discutida anteriormente.

2.3.2 Métodos baseados em inclinação

Holt [21] estendeu o SES para realizar previsões de séries com inclinação. Esse método, denominado de inclinação linear de Holt (*Holt's linear trend*), envolve uma equação de previsão e duas de suavização:

$$
\begin{aligned}
\text{Equação de previsão} \qquad & \hat{y}_{t+h|t} = \ell_t + hb_t \\
\text{Equação de nível} \qquad & \ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
\text{Equação de inclinação} \qquad & b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1},
\end{aligned}
$$

ℓ_t é o nível estimado da série no tempo t, b_t a inclinação da série temporal, α o parâmetro de suavização para o nível ($0 \leq \alpha \leq 1$) e β^* o parâmetro de suavização para a inclinação ($0 \leq \beta^* \leq 1$)

2.3.3 Métodos de inclinação amortizados

As previsões geradas pelo método de Holt crescem indefinidamente no futuro, levando a previsões superestimadas ou subestimadas. [23] introduziu um parâmetro que amortece a inclinação ao longo tempo. Métodos de inclinação amortizados (*Damped trend methods*) são um dos métodos mais utilizados para realizar previsões de múltiplas séries de forma automática.

Além de α e β^* introduzidos na Subseção 2.3.2, um parâmetro ϕ de amortecimento é utilizado:

$$
\begin{aligned}
\hat{y}_{t+h|t} &= \ell_t + (\phi + \phi^2 + \cdots + \phi^h)b_t \\
\ell_t &= \alpha y_t + (1-\alpha)(\ell_{t-1} + \phi b_{t-1}) \\
b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)\phi b_{t-1}.
\end{aligned}
$$

Com $0 < \phi < 1$. Se $\phi = 1$, então o método é idêntico a inclinação linear de Holt. Para valores entre 0 e 1, ϕ amortece a inclinação para uma constante quando t é grande o suficiente.

2.3.4 Método de Holt-Winters sazonal

[21] e [22] estenderam o método com sazonalidade, este modelo é chamado de Método de Holt-Winters sazonal (*Holt-Winters' seasonal method*). Adicionando um novo parâmetro s_t com o componente de suavização γ:

$$
\begin{aligned}
\text{Equação de previsão} \quad & \hat{y}_{t+h|t} = \ell_t + hb_t + s_{t+h-m(k+1)} \\
\text{Equação de nível} \quad & \ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
\text{Equação de inclinação} \quad & b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \\
\text{Equação de sazonalidade} \quad & s_t = \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}
\end{aligned}
$$

m é a frequência da sazonalidade, por exemplo: para dados trimestrais $m = 4$ e para dados mensais $m = 12$. k é a parte inteira de $\frac{h-1}{m}$.

2.3.5 Método de Holt-Winters sazonal amortizado

A versão amortizada do método de Holt-Winters sazonal (*Damped Holt-Winters' seasonal method*), é definida como:

$$
\begin{aligned}
\text{Equação de previsão} \quad & \hat{y}_{t+h|t} = \ell_t + \phi_h b_t + s_{t+h-m(k+1)} \\
\text{Equação de nível} \quad & \ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + \phi b_{t-1}) \\
\text{Equação de inclinação} \quad & b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)\phi b_{t-1} \\
\text{Equação de sazonalidade} \quad & s_t = \gamma(y_t - \ell_{t-1} - \phi b_{t-1}) + (1-\gamma)s_{t-m}
\end{aligned}
$$

2.3.6 Otimização

Todos os métodos desta seção possuem parâmetros que necessitam serem selecionados para a sua utilização, tais como α e β^*. Especialistas podem escolher estes parâmetros de forma subjetiva, mas uma maneira mais adequada é otimizar estes parâmetros de acordo com os valores observados. Desta forma, definindo o erro entre previsões como $e_t = y_t - \hat{y}_{t|t-1}$ com $t \in \{1, \ldots, T\}$, e considerando a série $y_1, \ldots, y_T$, minimizamos o seguinte problema de otimização não-linear

$$\sum_{t=1}^{T} (y_t - \hat{y}_{t|t-1})^2 = \sum_{t=1}^{T} e_t^2,$$

obtendo assim os parâmetros ótimos do método para uma dada série temporal.

2.3.7 Taxonomia de métodos de suavização exponencial

Dentro da taxonomia de modelos suavização exponenciais, a tabela 2.1 apresenta os nove tipos possíveis de combinações entre a utilização dos componentes de inclinação e sazonalidade. Entre eles, na Tabela 2.1 temos a suavização exponencial

Componente de inclinação	Componente sazonal		
	N (Nenhum)	A (Aditivo)	M (Multiplicativo)
N (Nenhum)	(N,N)	(N,A)	(N,M)
A (Aditivo)	(A,N)	(A,A)	(A,M)
A_d (Aditivo amortizado)	(A_d,N)	(A_d,A)	(A_d,M)

Tabela 2.1: Classificação de modelos de suavização exponencial

simples 2.3.1: (N,N); A inclinação linear de Holt 2.3.2: (A,N) e Holt-Winters sazonal amortizado 2.3.5: (A_d,A). Métodos com componentes multiplicativos foram desconsiderados neste trabalho.

2.4 ARIMA

Suavização exponencial e modelos ARIMA (definido formalmente posteriormente) são dois métodos muito utilizados e que possuem abordagens complementares para o problema de previsão de séries temporais. Métodos de suavização exponencial são baseados na inclinação e sazonalidade dos dados enquanto modelos ARIMA tem como base as autocorrelações da série temporal.

2.4.1 Modelos auto-regressivos

Em modelos de regressão, realizamos previsões de acordo com atributos relacionados às séries temporais. Em um modelo auto-regressivo, as previsões utilizam-se de valores passados da própria série temporal. A parte "auto"de auto-regressão significa que a regressão é realizada com uma variável sobre ela mesma.

Um modelo auto-regressivo de ordem p pode ser escrito como

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t,$$

no qual ε_t é uma variável aleatória com distribuição $\mathcal{N}(0, 1)$ e que representa um possível erro de previsão. c é uma constante e $\phi_1, \ldots, \phi_p$ parâmetros do modelo. Denotamos este modelo como modelo $AR(p)$, um modelo auto-regressivo de ordem p.

2.4.2 Modelos de média móvel

Ao invés de utilizar valores passados da série temporal como o modelo auto-regressivo, um modelo de média móvel utiliza erros passados para realizar uma "regressão".

$$y_t = c + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \cdots + \theta_q\varepsilon_{t-q},$$

Este modelo será denotado por MA(q), um modelo de média móvel de ordem q. Como os valores ε_t não são realmente observados, MA(q) não é uma regressão no sentido usual (uma regressão possuí todas as observações necessárias). Deste modo, para a seleção de $c, \phi_1, \ldots, \phi_p$ a estimativa de máxima verossimilhança (MLE) [24] é utilizada para minimizar

$$\sum_{t=1}^{T} \varepsilon_t^2.$$

MLE seleciona os mesmos parâmetros que a minimização de mínimos quadrados na regressão usual.

Uma distinção importante em relação a modelos de média móveis e métodos de suavização utilizando médias móveis: um modelo de média móvel é utilizado para prever futuros valores, enquanto a suavização é utilizada para estimar a inclinação e ciclo de valores passados.

2.4.3 Diferenciação

Diferenciação é uma maneira de estabilizar a média de uma série temporal pela remoção de variações do nível da série temporal e consequentemente reduzindo a inclinação e sazonalidade.

A série diferenciada é a variação entre valores consecutivos observados na série temporal original, definida como

$$y'_t = y_t - y_t.$$

A série diferenciada tem somente $T - 1$ valores, já que não é possível calcular y'_1 para a primeira observação.

Quando a série diferenciada é ruído branco, o modelo para a série original pode ser escrito como

$$y_t - y_{t-1} = \varepsilon_t,$$

onde $\varepsilon_t \sim \mathcal{N}(0, 1)$, ou seja, um ruído branco. Podemos escrever a equação acima da seguinte forma:

$$y_t = y_{t-1} + \varepsilon_t.$$

Definindo um modelo de passeio aleatório. Modelos de passeios aleatórios são

largamente utilizados para séries temporais que possuem longos períodos com tendência de subida ou descida, além de mudanças de direção.

Um modelo relacionado permite diferenças com médias diferentes de zero, ou seja,

$$y_t - y_{t-1} = c + \varepsilon_t \quad \text{or} \quad y_t = c + y_{t-1} + \varepsilon_t \, .$$

O valor de c é a média das variações consecutivas das observações. Se c é positivo, então a média da variação faz com que y_t tenha uma tendência de subida. Caso c seja negativo, o oposto ocorre e y_t tem uma tendência para baixo.

É possível diferenciar a série temporal mais de uma vez, uma série temporal diferenciada de segunda ordem é definida como

$$\begin{aligned} y''_t &= y'_t - y'_{t-1} \\ &= (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) \\ &= y_t - 2y_{t-1} + y_{t-2}. \end{aligned}$$

Nesse caso, y''_t pode ter no máximo $T - 2$ valores. A diferenciação de segunda ordem da série temporal modela uma variação de variações da série original.

2.4.4 Modelos ARIMA

Se combinarmos diferenciação (subtração entre valores consecutivos da série temporal) com auto-regressão e um modelo de média móvel, obtemos o modelo ARIMA (*AutoRegressive Integrated Moving Average*, usamos *Integrated* por ser o inverso da diferenciação) não-sazonal. O modelo é descrito por

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t, \tag{2.4}$$

No qual y'_t é o valor diferenciado da série. As observações do lado direito da equação incluem valores atrasados de y_t e valores atrasados de erro. Denotamos esse modelo ARIMA(p, d, q), no qual p é a parte auto-regressiva, d o grau de diferenciação e q a ordem da média móvel.

Diversos modelos clássicos são casos especiais de modelos ARIMA, alguns destes casos podem ser vistos na Tabela 2.2.

2.4.5 Interpretabilidade

Um aspecto importante de previsões é a interpretabilidade do modelo criado. Conforme podemos ver na Figura 2.2, a flexibilidade do modelo influência na sua inter-

Ruído branco	ARIMA(0,0,0) sem constante ($c = 0$)
Passeio aleatório	ARIMA(0,1,0) sem constante ($c = 0$)
Passeio aleatório com *drift*	ARIMA(0,1,0) com constante ($c \neq 0$)
Auto-regressão	ARIMA(p,0,0)
Média móvel	ARIMA(0,0,q)

Tabela 2.2: Casos especiais do modelo ARIMA. Fonte [4].

.

pretabilidade (Bias-Variance Tradeoff [1]). Em geral, quando a flexibilidade de um método de aprendizado estatístico aumenta, sua interpretabilidade diminui.

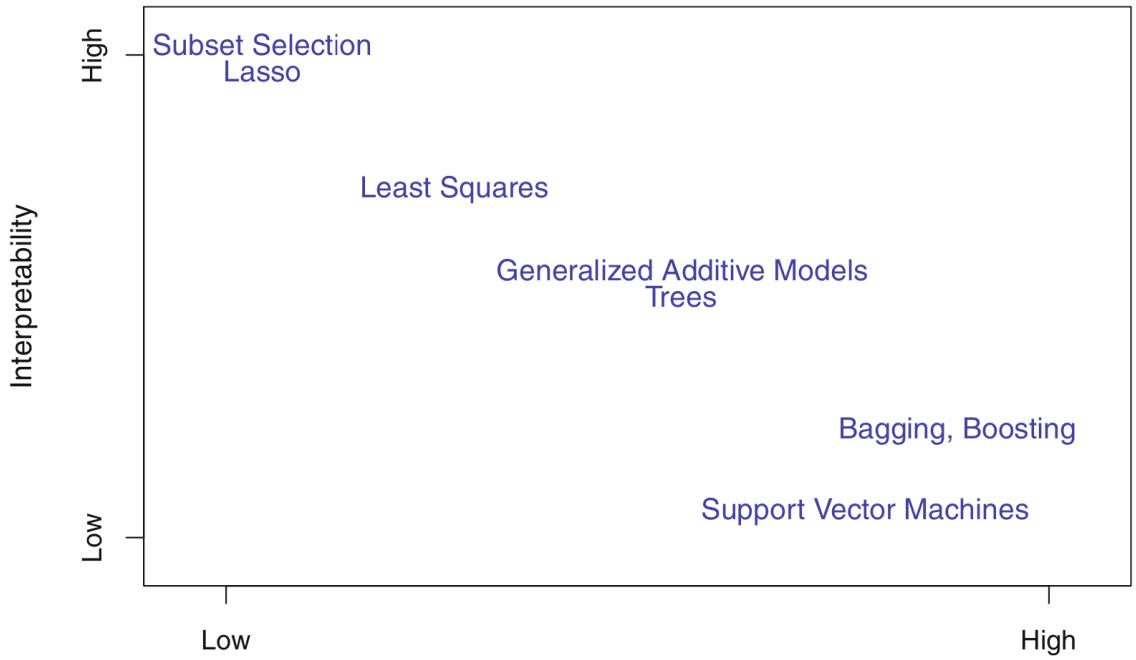


Figura 2.2: Balanceamento entre flexibilidade de um modelo e sua interpretabilidade. Fonte [1]

.

Neste trabalho ocorre uma perca de interpretabilidade dos métodos simples de previsão; para suavização exponencial; ARIMA até finalmente redes neurais. No Capítulo 3 veremos uma maneira de mitigar a perda de interpretabilidade através da quantificação do risco de previsões utilizando redes neurais probabilísticas.

Capítulo 3

Redes Neurais

Neste capítulo, apresentamos métodos totalmente automatizados e baseados em dados. Por exemplo, o modelo DeepAR (Seção 3.5), em contraste com os métodos apresentados no Capítulo 2, aprende prever sem precisar definir uma estrutura fixa para o modelo de previsão: a diferença de escala entre diferentes séries temporais de um conjunto de múltiplas séries; agrupamentos relevantes de séries temporais; problemas de *cold-start* em previsões de novas séries (por exemplo uma nova camisa de uma determinada categoria que conhecemos o comportamento geral); aprendizado de covariáveis relevantes como sazonalidades. Antes de apresentar o modelo DeepAR, técnicas de redes neurais será discutida brevemente.

3.1 *Single-layer perceptron*

O *Single-layer perceptron*, mais conhecido simplesmente por *Perceptron*, é um método supervisionado de aprendizado de máquina, um classificador binário desenvolvido em 1958 por Frank Rosenblatt [25] e pode ser descrito da seguinte maneira:

Dado uma entrada $\mathbf{x} = (x_1, x_2, \ldots, x_d)^\top$:

$$\text{Aprovar se:} \sum_{i=1}^{d} w_i x_i \geq \text{limiar}$$

$$\text{Negar se:} \sum_{i=1}^{d} w_i x_i < \text{limiar}$$

Isto pode ser expressado como uma fórmula linear $h \in \mathcal{H}$:

$$h(x) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - \text{limiar}\right)$$

Os fatores que alteram a hipótese são os valores w_i (pesos) e o limiar.

Concatenando uma constante artificial $x_0 = 1$ em $\mathbf{x}$ e definindo que o limiar estará representado pelo peso w_0, podemos simplificar esta fórmula para:

$$h(x) = \text{sign}(\mathbf{w} \bullet \mathbf{x})$$

onde $\bullet$ é o produto interno dos dois vetores $\mathbf{w}$ e $\mathbf{x}$, ou seja, $h(x) = \text{sign}\left(\left(\sum_{i=0}^{d} w_i x_i\right)\right)$; $\mathbf{x} = (x_0, x_1, \ldots, x_d)^\top$ são os componentes do vetor $\mathbf{x}$ ($x_0 = 1$); $\mathbf{w} = (w_0, w_1, \ldots, w_d)^\top$ (limiar $= w_0 = 1$). $h(x) = +1$ para créditos aprovados e $h(x) = -1$ para negados; $\text{sign}(s) = +1$ se $s > 0$ e $\text{sign}(s) = -1$ se $s \leq 0$.

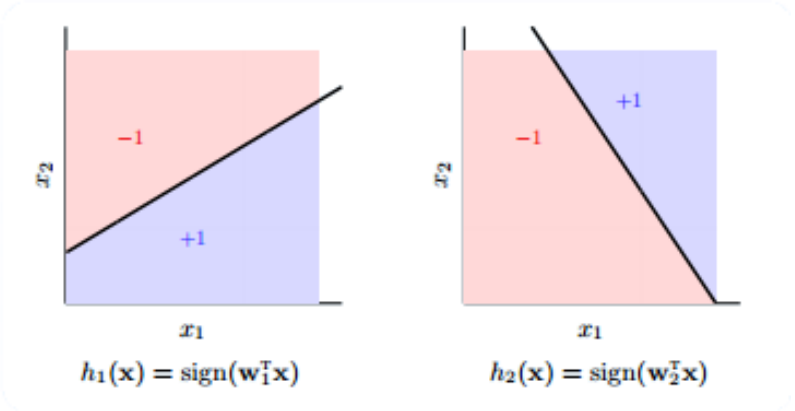


Figura 3.1: Exemplo de dois *perceptrons* com pesos diferentes. Fonte [2].

O ajuste de pesos para o aprendizado do *perceptron* é simples, para cada par $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$ do conjunto de dados classificado erroneamente ($\text{sign}(\mathbf{w}_t \bullet \mathbf{x}_t) \neq y_t$), (no passo $t \in \{0, 1, 2, \ldots\}$ do algoritmo), atualize os pesos da seguinte maneira: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_t \mathbf{x}_t$. O algoritmo termina quando não existe mais nenhum ponto classificado erroneamente no conjunto de dados, o que significa que o conjunto de dados é linearmente separável (para o caso que o conjunto de dados não é linearmente separável é necessário um limite no número de passos do algoritmo).

3.2 *Multi-layer perceptron*

Existem funções que um *perceptron* isolado não é capaz de generalizar, por exemplo a função XOR representada na Figura 3.2.

A solução para este problema é uma combinação de *perceptrons*, através de funções OR e AND é possível aproximar qualquer função arbitrária. As funções OR e AND podem ser representadas com 3 *perceptrons* cada: para o caso da função OR: $\mathbf{x} = (x_0 = 1, x_1, x_2)^\top$ e $\mathbf{w} = (w_0 = 1.5, w_1 = 1, w_2 = 1)^\top$; para a função AND: $\mathbf{x} = (x_0 = 1, x_1, x_2)^\top$ e $\mathbf{w} = (w_0 = -1.5, w_1 = 1, w_2 = 1)^\top$. Podemos representar graficamente estas funções de acordo com a Figura 3.3.

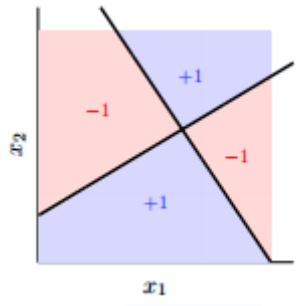


Figura 3.2: Função XOR: não é possível achar uma combinação de pesos para o perceptron que separe a função linearmente. Fonte [2].

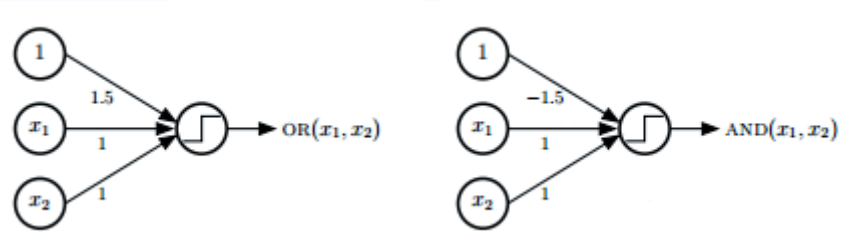
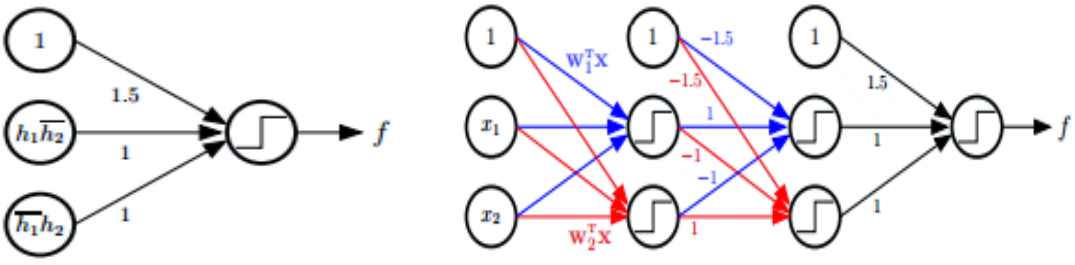


Figura 3.3: União de funções OR e AND. Fonte [2].



(a) Exemplo de função complexa. Fonte [2].

(b) *Multi-layer perceptron.* Fonte [2].

Figura 3.4

Um exemplo de função complexa arbitrária é representada na Figura 3.4a, esta função arbitrária pode ser implementada através da combinação de OR's e AND's e é dada pela Figura 3.4b.

Multi-layer perceptrons podem aproximar qualquer função pelo teorema de aproximação universal [26], a Figura 3.5 mostra a área interna de um círculo sendo aproximado por uma combinação de *perceptrons*.

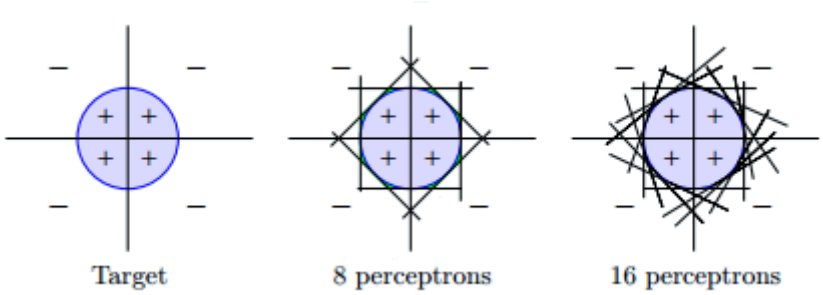


Figura 3.5: Função para aproximar a área interna de um círculo. Fonte [2].

3.3 Redes neurais

Redes neurais, ilustrada na Figura 3.6, são uma forma de *Multi-layer perceptron* no qual os *perceptrons* são substituídos por funções não-lineares, por exemplo $h(x) = \tanh(\mathbf{w} \bullet \mathbf{x})$, estes novos componentes são chamados de neurônios e suas saídas são interpretadas como funções de ativação devido a sua semelhança com sinapses cerebrais.

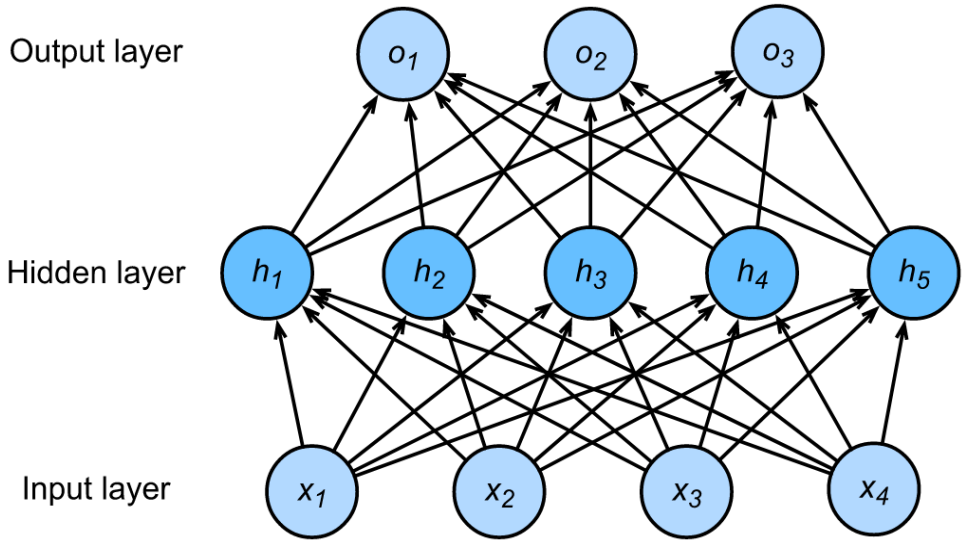


Figura 3.6: Rede neural. Fonte [3].

Redes neurais começaram a ganhar popularidade quando se mostrou ser possível resolver problemas anteriormente insolúveis com outras técnicas de otimização

da rede [27]. O artigo descreve o *backpropagation*, um algoritmo que otimiza os parâmetros da rede de pesos de uma forma muito mais eficiente do que métodos apresentados anteriormente.

Formalmente, uma rede com uma única camada oculta pode ser calculada da seguinte maneira para uma observação $\mathbf{x} \in \mathbb{R}^{1\times d}$:

$$\begin{aligned}\mathbf{h} &= \phi(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1),\\ \mathbf{o} &= \mathbf{W}_2\mathbf{h} + \mathbf{b}_2.\end{aligned}$$

onde $\mathbf{o} \in \mathbb{R}^{1\times q}$ é variável de saída, $\mathbf{W}_1$ e $\mathbf{W}_2 \in \mathbb{R}^{h\times q}$ os parâmetros de peso e $\mathbf{b}_1$ e $\mathbf{b}_2 \in \mathbb{R}^{1\times q}$ o parâmetro de viés. A função de ativação da camada escondida é ϕ (por exemplo, a função $\phi = \tanh$)

Também podemos aumentar o número de camadas através da conexão entre camadas ocultas, por exemplo: $\mathbf{h}_1 = \phi(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$, $\mathbf{h}_2 = \phi(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)$ e $\mathbf{o} = \mathbf{W}_2\mathbf{h}_2 + \mathbf{b}_2$.

Para melhorar a eficiência computacional, tipicamente calculamos as operações da rede neural com *minibatches* de dados. Um *minibatch* $\mathbf{X}$ de exemplos possui n exemplos e tem dimensionalidade d, $1 < n <$ #(conjunto de treinamento). Assumimos que a saída possui q categorias. Logo, o *minibatch* $\mathbf{X}$ tem dimensões $\mathbb{R}^{n\times d}$, pesos $\mathbf{W} \in \mathbb{R}^{d\times q}$ e viéses $\mathbf{b} \in \mathbb{R}^q$.

Vetorizando as equações anteriores para um *minibatch* $\mathbf{X}$ (ao invés de uma entrada), obtemos

$$\begin{aligned}\mathbf{H}_1 &= \phi(\mathbf{W}_1\mathbf{X} + \mathbf{b}_1),\\ \mathbf{H}_2 &= \phi(\mathbf{W}_2\mathbf{H}_1 + \mathbf{b}_2),\\ \mathbf{O} &= \mathbf{W}_3\mathbf{H}_2 + \mathbf{b}_3.\end{aligned} \tag{3.1}$$

O objetivo do treinamento da rede é otimizar uma função de custo (*loss function*) $l(y, o_i)$ (para uma única amostra). Por exemplo, para a raiz do erro médio quadrático, a função de custo pode ser calculada para um conjunto de dados da seguinte forma

$$L = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log o_i - \log \tilde{y}_i)^2},$$

Com a função de custo definida podemos aplicar o algoritmo de *backpropagation* que será apresentada na Subseção 3.4.2. A próxima seção define de forma mais detalhada as equações acima para o problema de séries temporais, foco deste trabalho, incluíndo a versão do algoritmo de *backprogation* para séries temporais.

3.4 Redes neurais recorrentes

Uma maneira de modelar $p(x_t \mid x_{t-1}, \ldots, x_{t-n+1})$ sem armazenar toda a informação da distribuição condicional é utilizar um modelo de variável latente no qual

$$p(x_t \mid x_{t-1}, \ldots, x_1) \approx p(x_t \mid h_{t-1}).$$

onde h_t é uma variável latente que armazena a informação da sequência. Uma variável latente também é chamada de variável escondida (*hidden variable*) ou estado escondido (*hidden state*). A variável latente no tempo t pode ser calculada com base na entrada x_t e na variável latente h_{t-1}:

$$h_t = z(x_t, h_{t-1}).$$

Para uma função z suficientemente complexa, z deixa de ser uma aproximação. Por exemplo, z poderia guardar todas as observações da série temporal. Porém quanto mais complexa a função z maior custo computacional e de memória.

Redes neurais recorrentes são redes neurais com variáveis latentes.

3.4.1 Redes neurais recorrentes

Seja t o passo da iteração do treinamento, que define o *minibatch* sendo considerado. $\mathbf{X}_t \in \mathbb{R}^{n \times d}$ é o *minibatch* escolhido no tempo $t = 1, \ldots, T$ de uma iteração, onde n é o tamanho da amostra selecionada e d a dimensão do vetor de entrada das observações. $\mathbf{H}_t \in \mathbb{R}^{n \times h}$ é a variável escondida do passo t da sequência. Guardamos a variável escondida $\mathbf{H}_{t-1}$ do passo anterior e introduzimos o parâmetro de peso $\mathbf{W}_{hh} \in \mathbb{R}^{h \times h}$ no modelo de redes neurais (sem utilização de variáveis latentes), desta forma conseguimos descrever como utilizar as informações da variável latente no passo anterior no tempo atual.

$$\mathbf{H}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh} + \mathbf{H}_{t-1} \mathbf{W}_{hh} + \mathbf{b}_h)$$

Comparado com a Equação (3.1), adicionamos o termo $\mathbf{H}_{t-1}\mathbf{W}_{hh}$ para capturar o relacionamento entre os elementos da sequência até o passo atual da iteração, sendo a variável latente da rede neural. Como a variável latente usa a mesma definição do passo anterior para o passo atual, esta é uma equação recorrente, logo o nome redes neurais recorrentes (RNN). Para o passo t, a saída da camada de saída é

$$\mathbf{O}_t = \mathbf{H}_t \mathbf{W}_{hq} + \mathbf{b}_q.$$

Os parâmetros da RNN são os pesos $\mathbf{W}xh \in \mathbb{R}^{d \times h}, \mathbf{W}_{hh} \in \mathbb{R}^{h \times h}$ da camada

oculta com viés $\mathbf{b}_h \in \mathbb{R}^{1\times h}$ e o peso $\mathbf{W}_{hq} \in \mathbb{R}^{h\times q}$ com viés $\mathbf{b}_q \in \mathbb{R}^{1\times q}$. Os mesmos parâmetros são utilizados em cada passo t, logo o número de parâmetros é independente do número de passos em uma iteração da RNN.

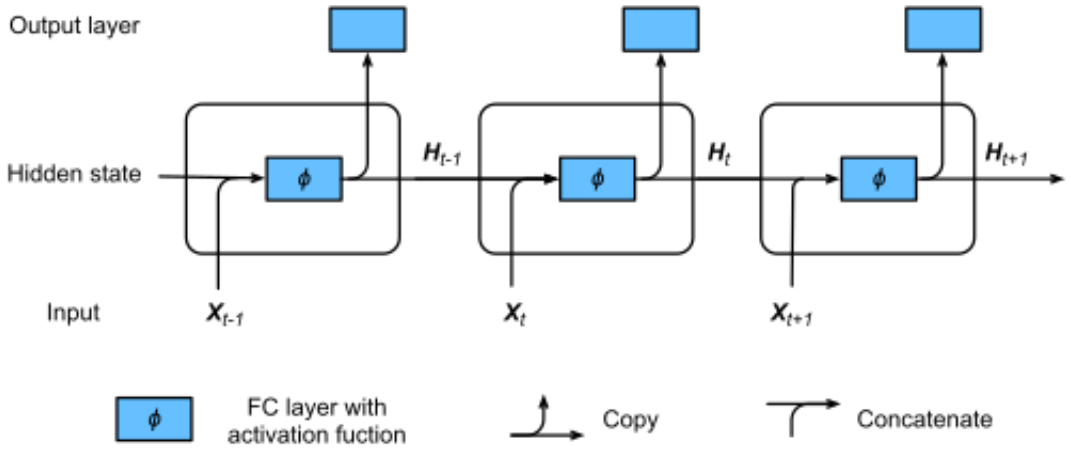


Figura 3.7: RNN com uma camada escondida. Fonte [3].

A Figura 3.7 mostra a lógica de computação de uma RNN em três passos consecutivos. No passo t, o cálculo da camada escondida pode ser tratada como uma entrada de uma camada totalmente conectada com uma função de ativação ϕ depois de concatenar a entrada $\mathbf{X}_t$ com a camada escondida $\mathbf{H}_{t-1}$ do passo anterior. A saída da camada totalmente conectada é a camada escondida do passo atual $\mathbf{H}_t$. Os parâmetros do modelo são $\mathbf{W}_{xh}$ e $\mathbf{W}_{hh}$, com viés $\mathbf{b}_h$. $\mathbf{H}_t$ também é entrada para $\mathbf{O}_t$, a camada totalmente conectada de saída do passo atual.

3.4.2 *Backpropagation through time*

Backpropagation through time (BPTT) é basicamente a aplicação de backpropagation para modelos de sequência com um estado oculto.

Se decompormos $\mathbf{W}$ nas diferentes matrizes de peso ($\mathbf{W}_{hx}, \mathbf{W}_{hh}$ e $\mathbf{W}_{oh}$), temos um simples modelo linear latente:

$$\mathbf{h}_t = \mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} \text{ e } \mathbf{o}_t = \mathbf{W}_{oh}\mathbf{h}_t.$$

Calculando os gradientes $\frac{\partial L}{\partial \mathbf{W}hx}$, $\frac{\partial L}{\partial \mathbf{W}hh}$ e $\frac{\partial L}{\partial \mathbf{W}_{oh}}$ para

$$L(\mathbf{x}, \mathbf{y}, \mathbf{W}) = \sum_{t=1}^{T} l(\mathbf{o}_t, y_t),$$

no qual $l(\cdot)$ é uma função de perda (*loss function*). Derivando com respeito a W_{oh}, obtemos

$$\partial_{\mathbf{W}_{oh}} L = \sum_{t=1}^{T} \Phi\left(\partial_{\mathbf{o}_t} l(\mathbf{o}_t, y_t), \mathbf{h}_t\right),$$

$\Phi(\cdot)$ denota a multiplicação (convencional) entre duas ou mais matrizes. Derivando para $\mathbf{W}_{hx}$ e $\mathbf{W}_{hh}$ temos

$$\partial_{\mathbf{W}_{hh}} L = \sum_{t=1}^{T} \Phi\left(\partial_{\mathbf{o}_t} l(\mathbf{o}_t, y_t), \mathbf{W}_{oh}, \partial_{\mathbf{W}_{hh}} \mathbf{h}_t\right),$$

$$\partial_{\mathbf{W}_{hx}} L = \sum_{t=1}^{T} \Phi\left(\partial_{\mathbf{o}_t} l(\mathbf{o}_t, y_t), \mathbf{W}_{oh}, \partial_{\mathbf{W}_{hx}} \mathbf{h}_t\right).$$

$$\partial_{\mathbf{h}_t} \mathbf{h}_{t+1} = \mathbf{W}_{hh}^{\top}, \text{ logo } \partial_{\mathbf{h}_t} \mathbf{h}_T = \left(\mathbf{W}_{hh}^{\top}\right)^{T-t}.$$

Juntando os termos obtemos

$$\partial_{\mathbf{W}_{hh}} \mathbf{h}_t = \sum_{j=1}^{t} \left(\mathbf{W}_{hh}^{\top}\right)^{t-j} \mathbf{h}_j$$

$$\text{e } \partial_{\mathbf{W}_{hx}} \mathbf{h}_t = \sum_{j=1}^{t} \left(\mathbf{W}_{hh}^{\top}\right)^{t-j} \mathbf{x}_j.$$

3.4.3 *Long Short Term Memory* (LSTM)

Embora uma RNN possa ser tão complexa quanto desejarmos, resolver problemas reais com RNNs como descritas na Seção 3.4 sofrem de instabilidades numéricas no cálculo dos gradientes, fazendo que os gradientes sofram *overflow* ou *underflow* [28]. Uma das primeiras abordagens para resolver este problema foi a criação da LSTM [29] (atualmente o artigo mais citado do século XX em aprendizado profundo, segundo o Google Scholar). Três portões são introduzidos em LSTMs: o portão de entrada (*input gate*), o portão de esquecimento (*forget gate*) e o portão de saída (*output gate*). Além disso, é adicionada uma célula de memória (*memory cell*). Todas essas alterações tornam o estado escondido mais sofisticado com o propósito de guardar informações adicionais, como o desafio de utilizar informações relevantes de longo prazo e descartar informações que não são importantes de curto prazo.

Como na seção anterior: h é o número de unidades escondidas, n é o tamanho do *minibatch* e d a dimensão de uma observação. Logo, a entrada é $\mathbf{X}_t \in \mathbb{R}^{n \times d}$. O estado oculto do último passo é $\mathbf{H}_{t-1} \in \mathbb{R}^{n \times h}$. Os portões são definidos da seguinte maneira: portão de entrada $\mathbf{I}_t \in \mathbb{R}^{n \times h}$, portão de esquecimento $\mathbf{F}_t \in \mathbb{R}^{n \times h}$, portão de saída $\mathbf{O}_t \in \mathbb{R}^{n \times h}$. O cálculo dos portões são

$$\mathbf{I}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xi} + \mathbf{H}_{t-1}\mathbf{W}_{hi} + \mathbf{b}_i),$$
$$\mathbf{F}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xf} + \mathbf{H}_{t-1}\mathbf{W}_{hf} + \mathbf{b}f),$$
$$\mathbf{O}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xo} + \mathbf{H}_{t-1}\mathbf{W}_{ho} + \mathbf{b}_o),$$

$\mathbf{W}_{xi}, \mathbf{W}_{xf}, \mathbf{W}_{xo} \in \mathbb{R}^{d\times h}$ e $\mathbf{W}_{hi}, \mathbf{W}_{hf}, \mathbf{W}_{ho} \in \mathbb{R}^{h\times h}$ são parâmetros de peso e $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o \in \mathbb{R}^{1\times h}$ são o viés.

Uma candidata para a célula de memória é dado por $\tilde{\mathbf{C}}_t \in \mathbb{R}^{n\times h}$ e calculada como

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{X}_t\mathbf{W}_{xc} + \mathbf{H}_{t-1}\mathbf{W}_{hc} + \mathbf{b}_c).$$

tanh faz com que o valor de $\tilde{\mathbf{C}}_t$ fique entre $[-1, 1]$. $\mathbf{W}_{xc} \in \mathbb{R}^{d\times h}$ e $\mathbf{W}_{hc} \in \mathbb{R}^{h\times h}$ são parâmetros de peso e $\mathbf{b}_c \in \mathbb{R}^{1\times h}$ um parâmetro de viés.

$\mathbf{I}_t$ controla quanto de $\tilde{\mathbf{C}}_t$ introduzimos na célula de memória e $\mathbf{F}_t$ o quanto esquecemos da memória $\mathbf{C}_{t-1} \in \mathbb{R}^{n\times h}$ do passo anterior.

$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tilde{\mathbf{C}}_t.$$

onde $\odot$ é uma multiplicação elemento a elemento. Finalmente, a saída da camada oculta é calculada como

$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh(\mathbf{C}_t).$$

A Figura 3.8 mostra o fluxo dos dados de acordo com as equações definidas anteriormente.

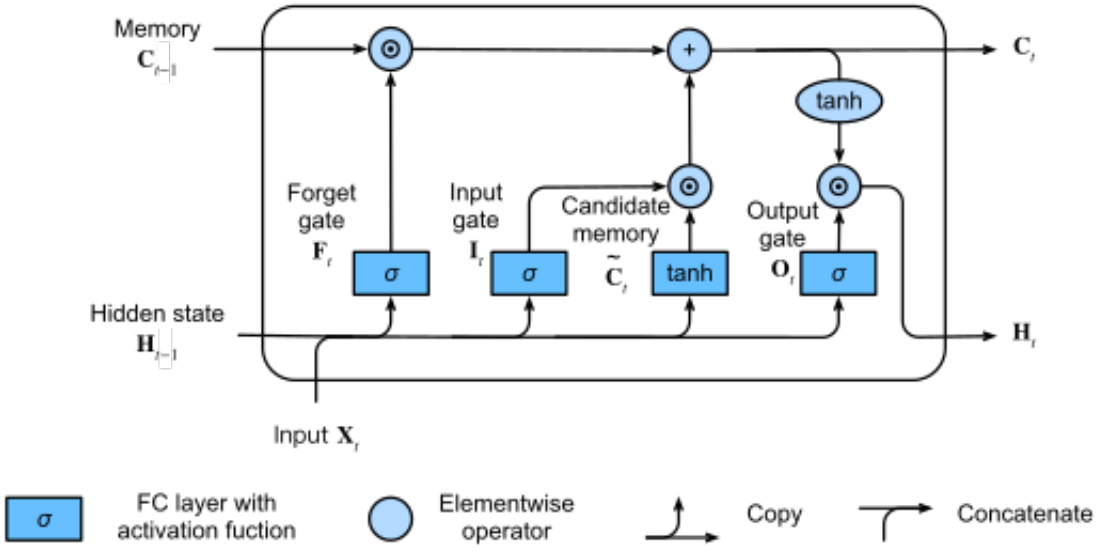


Figura 3.8: Cálculo do estado escondido. As multiplicações são elemento a elemento. Fonte [3].

3.5 DeepAR

DeepAR é um modelo baseado em redes neurais recorrentes auto-regressivas, que aprende um modelo global para todas as séries temporais do conjunto de dados (em oposição ao treinamento de um modelo por série temporal). DeepAR tem uma arquitetura de RNN para previsões probabilísticas e tratamento especial para os casos que as séries temporais possuem magnitudes muito diferentes.

Algumas características importantes do DeepAR são:

1. Enquanto o modelo aprende a sazonalidade e dependências entre os atributos, comportamentos de grupo são capturados com necessidade mínima de tratamento manual dos dados;

2. DeepAR faz previsões probabilísticas na forma de amostras de Monte Carlo que podem ser usados para calcular estimativas de quantis para todo o horizonte das séries;

3. O comportamento de itens similares é capturado, sendo possível realizar previsões para séries com pouco ou nenhum dado anterior;

4. Modelos normalmente assumem ruídos gaussianos, a escolha da função de verossimilhança é uma opção do usuário no DeepAR, podendo escolher uma função apropriada de acordo com as características dos dados.

As características 1 e 3 diferenciam o DeepAR de modelos clássicos de previsão, as características 2 e 4 dizem respeito a produzir distribuições de previsões acuradas aprendidas do comportamento de todas as séries temporais de forma conjunta. Previsões probabilísticas permitem a otimização de funções de risco de acordo com a incerteza associada a distribuição da previsão.

Denotamos o valor da série temporal i no tempo t por $y_{i,t}$, o objetivo é modelar a distribuição condicional

$$P(\mathbf{y}_{i,t_0:T}|\mathbf{y}_{i,1:t_0-1}, \mathbf{x}_{i,1:T})$$

do futuro de cada série temporal $[\mathbf{y}_{i,t_0}, \mathbf{y}_{i,t_0+1}, \ldots, \mathbf{y}_{i,T}] \coloneqq \mathbf{y}_{i,t_0:T}$, t_0 denota o instante de tempo que assumimos não mais conhecer $y_{i,t}$ no momento de predição. $\mathbf{x}_{i,1:T}$ são atributos conhecidos em todos os pontos no tempo, $1, \ldots, T$. O modelo resumido pela Figura 3.9 é baseado em uma arquitetura de rede neural auto-regressiva. Assumindo que a distribuição do modelo segue $Q_\Theta(\mathbf{y}_{i,t_0:T}|\mathbf{y}_{i,1:t_0-1}\mathbf{x}_{i,1:T})$ que pode ser decomposta em um produto de fatores de verossimilhanças

$$Q_\Theta(\mathbf{y}_{i,t_0:T}|\mathbf{y}_{i,1:t_0-1}, \mathbf{x}_{i,1:T}) = \prod_{t=t_0}^{T} Q_\Theta(y_{i,t}|\mathbf{y}_{i,1:t_0-1}, \mathbf{x}_{i,1:T}) = \prod_{t=t_0}^{T} \ell(y_{i,t}|\theta(\mathbf{h}_{i,t}, \Theta))$$

parametrizada pela saída do estado escondido $\mathbf{h}_{i,t}$, no tempo t da série i, de uma rede neural recorrente auto-regressiva

$$\mathbf{h}_{i,t} = g(\mathbf{h}_{i,t-1}, \mathbf{y}_{i,t-1}, \mathbf{x}_{i,t}, \Theta), \tag{3.2}$$

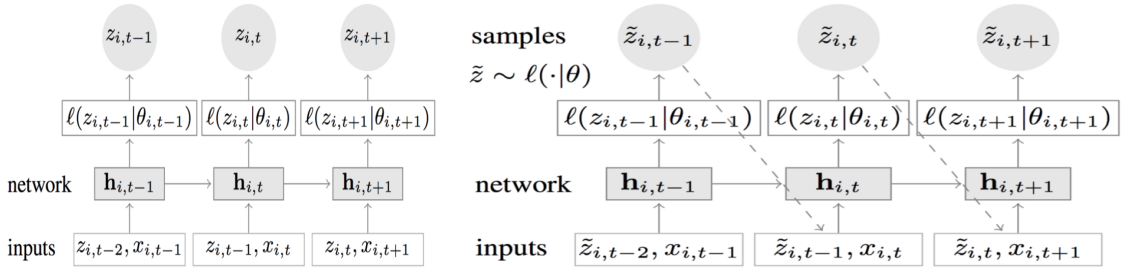
onde g é uma função implementada por uma rede neural recorrente com células LSTM. Note que o modelo também é auto-regressivo utilizando a saída $\mathbf{y}_{i,t-1}$ do passo anterior como entrada. A função de verossimilhança $\ell(y_{i,t}|\theta(\mathbf{h}_{i,t}))$ é uma distribuição de probabilidade fixa cujo parâmetros são dados pela função $\theta(\mathbf{h}_{i,t}, \Theta)$. A função de verossimilhança determina o "modelo de ruído"e deve ser escolhido de tal forma que corresponda as propriedades estatísticas das séries temporais. Neste trabalho, utilizamos a verossimilhança gaussiana para valores pertencentes a $\mathbb{R}$ (embora outras verossimilhanças possam ser utilizadas, desde que amostras dessas distribuições possam ser facilmente obtidas). A parametrização da verossimilhança gaussiana é dada por sua média e desvio padrão, $\theta = (\mu, \sigma)$, onde a média é dada por uma função afim da saída da rede e o desvio padrão é obtido pela transformação afim seguida de uma ativação com a função *softplus* (descrita a seguir) para garantir que $\sigma > 0$:

$$\ell_G(z|\mu, \sigma) = (2\pi\sigma^2)^{\frac{-1}{2}} \exp\left(-(z-\mu)^2/(2\sigma^2)\right)$$

$$\mu(\mathbf{h}_{i,t}) = \mathbf{w}_\mu^\top \mathbf{h}_{i,t} + b_\mu \text{ e } \sigma(\mathbf{h}_{i,t}) = \log(1 + \exp\left(\mathbf{w}_\sigma^\top \mathbf{h}_{i,t} + b_\sigma\right))$$

Informações das observações, no intervalo da distribuição condicional $\mathbf{y}_{i,1:t_0-1}$, são transferidas para o intervalo de predição através do estado inicial $\mathbf{h}_{i,t_0-1}$. Nessa configuração de *sequence-to-sequence*, é utilizado uma rede codificadora (*encoder*) para o intervalo de condicionamento e uma rede decodificadora (*decoder*) para o intervalo de predição. O estado inicial é a saída da rede codificadora. Embora as arquiteturas da rede codificadora e decodificadora possam ser diferentes, neste trabalho as arquiteturas são iguais. Além disso, os pesos entre as redes são compartilhadas, de tal forma que o estado inicial da rede decodificadora $\mathbf{h}_{i,t_0-1}$ é obtido através do cálculo da Equação 3.2 para $t = 1, \ldots, t_0 - 1$, onde já ocorreram as observações. O estado inicial da rede codificadora $\mathbf{h}_{i,0}$ e $y_{i,0}$ é inicializado com zeros.

Podemos ver na Figura 3.9a o treinamento do modelo: Em cada passo t, as entradas da rede são os atributos $x_{i,t}$, o valor da série no passo anterior $y_{i,t-1}$ e o



(a) Rede em tempo de treinamento. Fonte [19].

(b) Rede em tempo de predição. Fonte [19].

Figura 3.9: Sumário do modelo. z e $\tilde{z}$ correspondem a y e $\tilde{y}$ no texto, respectivamente. Repare que a mesma rede é utilizada por todas as séries temporais e a mesma arquitetura para o treinamento é utilizada em tempo de predição. A série temporal $y_{i,t}$ é alimentada ao modelo para $t < t_0$, então no tempo de predição ($t \geq 0$, Figura 3.9b), uma amostra $\tilde{y}_{i,t} \sim \ell(\cdot|\theta_{i,t})$ é dada como entrada para o próximo passo da rede e repetido esse processo até o fim do intervalo de predição $t = t_0 + T$ gerando um único traço da amostra. A repetição desse processo de predição gera diversos traços gerando assim a distribuição conjunta empírica de predição.

estado escondido $\mathbf{h}_{i,t-1}$. Então $\mathbf{h}_{i,t} = g(\mathbf{h}_{i,t-1}, \mathbf{y}_{i,t-1}, \Theta)$ é utilizado para calcular os parâmetros $\theta(\mathbf{h}_{i,t}, \Theta)$ da verossimilhança $\ell(y|\theta)$, estes parâmetros são então usados para treinar o modelo através da minimização de

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{t=t_0}^{T} \log \ell(y_{i,t}|\theta(\mathbf{h}_{i,t})). \tag{3.3}$$

Dado os parâmetros do modelo Θ, nós podemos obter amostras da distribuição conjunta $\tilde{\mathbf{y}}_{i,t_0:T} \sim Q_\Theta(\mathbf{y}_{i,t_0:T}|\mathbf{y}_{i,1:t_0-1}\mathbf{x}_{i,1:T})$ através de amostragem ancestral (*ancestral sampling*): Primeiro, obtemos $\mathbf{h}_{i,t_0-1}$ calculando a Equação 3.2 para $t = 1, \ldots, t_0$. Para $t = t_0, t_0 + 1, \ldots, T$ amostramos $\tilde{y} \sim \ell(\cdot|\theta(\tilde{\mathbf{h}}_{i,t}, \Theta))$ no qual $\tilde{\mathbf{h}}_{i,t} = g(\mathbf{h}_{i,t-1}, \tilde{\mathbf{y}}_{i,t-1}, \mathbf{x}_{i,t}, \Theta)$ é inicializado com $\tilde{\mathbf{h}}_{i,t-1} = \mathbf{h}_{i,t-1}$ e $\tilde{y}_{i,t_0-1} = y_{i,t_0-1}$. Amostras obtidas do modelo desta forma podem então serem usadas para calcular medidas de interesse, como quantis da distribuição da soma dos valores para algum intervalo no futuro.

Um problema para o treinamento da rede é que tanto a entrada auto-regressiva $y_{i,t-1}$ como a saída da rede (μ ou σ) seguem diretamente a magnitude da observação $y_{i,t}$, mas as não-linearidades da rede possuem um intervalo limitado de saída. Para resolver o tratamento da escala das séries temporais, sem modificações no método, a rede necessita aprender a ajustar a entrada para um intervalo apropriado na camada de entrada e então inverter esse ajuste na camada de saída. Então as entradas auto-regressivas ($y_{i,t}$ ou $\tilde{y}_{i,t}$) são divididas por um fator dependente da série ν_i. Da mesma maneira os parâmetros μ ou σ da função de verossimilhança são multiplicado por

ν_i. Uma heurística que costuma funcionar na prática é definir $\nu_i = 1 + \frac{1}{t_0}\sum_{t=1}^{t_0} y_{i,t}$.

Um outro problema é que devido ao desequilíbrio do volume das séries temporais, um método de otimização estocástica que amostra séries temporais de maneira uniforme visitará infrequentemente séries temporais com alto volume, o que significa um provável *underfitting* dessas séries temporais. Isso é especialmente problemático no caso de previsão de demanda, no qual itens muito vendidos tem comportamentos diferentes de itens pouco vendidos (como veremos no Capítulo 5) e ter previsões para itens muito vendidos é mais importante em termos de negócio. Para combater este problema, é realizada uma amostragem de maneira não-uniforme durante o treinamento, no qual a probabilidade de selecionar um intervalo com magnitude ν_i é proporcional a ν_i. Este esquema de amostragem é simples, porém efetivo.

As covariáveis $\mathbf{x}_{i,t}$ podem ser dependentes do item, dependentes do tempo, ou ambas. Dois exemplos de informações adicionais para a série temporal são informações de mercado (IBOVESPA) ou a semana do ano que podem ser usadas como entradas para o modelo. As covariáveis também podem ser usadas para incorporar informações que podem influenciar o resultado, uma promoção ativa de um produto ou preço são exemplos de covariáveis deste tipo. Importante notar que estas covariáveis precisam ser conhecidas em tempo de predição, $t \geq t_0$. Algumas covariáveis são adicionadas automaticamente a todos os experimentos: uma covariável do tempo entre a primeira observação e a observação atual; a hora do dia; dia da semana; semana do ano e mês do ano. Além disso, variáveis categóricas para identificação de padrões específicos das séries são adicionadas, como a categoria do produto: roupa, camisa, tipo de geladeira, etc. Essas informações são padronizadas para terem média zero e variância unitária e são aprendidas como um *embedding* do modelo.

Em alguns casos, os valores $y_{i,t}$ podem não terem sido observados ou não estarem presentes para uma parcela da série temporal. Por exemplo, no contexto de previsão de demanda, um produto pode ter ficado fora de estoque por um certo período no qual a demanda deste produto não foi observada. Não modelar as observações faltantes pode levar o modelo a ter um viés abaixo do real e, no pior caso, fazer com que produtos fora de estoque tenham como consequência previsões menores, menos reabastecimentos e repetindo o ciclo de mais situações fora de estoque. No DeepAR, observações faltantes são tratadas de uma maneira fundamentada com a substituição de cada valor não observado $y_{i,t}$ por uma amostra $\tilde{y}_{i,t} \sim \ell(\cdot|\theta(\mathbf{h}_{i,t}))$ da distribuição condicional preditiva no cálculo da Equação 3.2, desconsiderando o termo de verossimilhança da observação faltante correspondente na Equação 3.3.

Capítulo 4

Avaliação de acurácia das previsões

Todos os modelos de previsão possuem parâmetros que devem ser ajustados utilizando dados disponíveis. Entretanto, este mesmo dado deve ser utilizado para avaliar a acurácia dos modelos uma vez parametrizados. Dessa forma, é importante avaliar o comportamento das previsões em dados que não foram utilizados na fase de parametrização do modelo, conhecida por fase de treinamento, e assim podemos avaliar seu desempenho para quando o modelo realizar previsões fora da amostra. Para este fim, é comum separar as séries temporais em duas partes, uma utilizada na fase de treinamento e outra na de teste (Figura 4.1). A parte de treinamento é utilizada para estimar os parâmetros e a parte de teste para avaliar a acurácia do modelo parametrizado.

Figura 4.1: Divisão entre conjunto de treino e teste. Fonte [4].

É importante notar que um modelo que possui bom desempenho no conjunto de treinamento não necessariamente realizará previsões melhores que um modelo que possui maior erro na fase de treinamento. Dado um número grande o suficiente de parâmetros sempre é possível ter erro zero na parte de treinamento, conhecido por sobre-ajuste (*overfitting*). Realizar um sobre-ajuste do modelo aos dados é tão ruim quanto não conseguir identificar o padrão existente nos dados.

4.1 Erro de previsão

Um erro de previsão é definido como a diferença entre o valor observado e sua previsão, formalmente:

$$e_{T+h} = y_{T+h} - \hat{y}_{T+h|T},$$

onde $\hat{y}_{T+h|T}$ é o valor da previsão dado que o modelo observa as T amostras anteriores, y_{T+h} é o valor observado, e h é o passo futuro da previsão. A parte de treinamento é dada por $\{y_1, \ldots, y_T\}$ e teste por $\{y_{T+1}, y_{T+2}, \ldots\}$.

4.2 Erros dependentes de escala

Duas medidas clássicas de erro são o Erro Médio Absoluto (*mean absolute error* (MAE)) e a Raiz do Erro Médio Quadrático (*Root mean squared error* (RMSE)), definidas da seguinte forma:

$$\text{Erro médio absoluto: MAE} = \frac{1}{N}\sum_{t=1}^{N}|e_t|,$$

$$\text{Raiz do erro médio quadrático: RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(e_t^2)}.$$

onde N é igual o número de pontos da série temporal.

Medidas de erros que usam somente e_t não devem ser utilizadas para comparar diferentes séries temporais, pois magnitudes diferentes entres as séries levam a magnitudes diferentes de erros, o que fazem as comparações terem menos significado.

4.3 Erros independentes de escala

Erros relativos tem a vantagem de serem livres de unidade, podendo ser usado para comparar o desempenho de diferentes séries temporais (ou conjunto de dados). A métrica mais comum é a Média Percentual Absoluta do Erro (*Mean Absolute Percentage Error*, MAPE):

$$\text{MAPE} = \frac{1}{N}\sum_{t=1}^{N}\left|100\frac{e_t}{y_t}\right|.$$

Alguns problemas existem ao usar o MAPE, se $y_t = 0$ para algum t, o valor do MAPE é indefinido. Se y_t tende a zero, o MAPE tende a infinito. Uma outra desvantagem do MAPE é que penalidades para erros negativos são maiores que penalidades para erros positivos. Uma versão simétrica do MAPE (sMAPE) foi proposta em [30], definida por

$$\text{sMAPE} = \frac{1}{N}\sum_{t=1}^{N}200\frac{|e_t|}{y_t + \hat{y}_t}.$$

4.4 Validação cruzada de séries temporais

Uma maneira de melhor avaliar o comportamento de um modelo fora da amostra é a utilização de validação cruzada. Neste caso, diversas séries temporais são criadas a partir da série temporal original, com cada conjunto de teste possuindo uma única observação fora do respectivo conjunto de treino. Cada conjunto de treino possui somente observações que ocorreram anteriormente a observação do conjunto de teste. A Figura 4.2 ilustra as séries de treino e teste (azul - treino, vermelho - teste), construídas a partir de uma única série temporal.

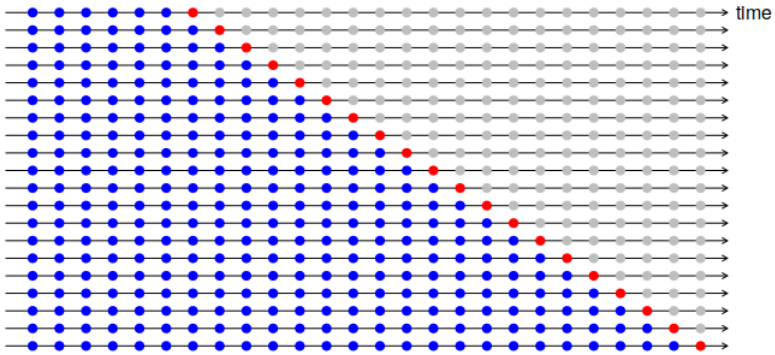


Figura 4.2: Avaliação deslizante com um passo a frente. Fonte [4].

A acurácia do modelo é então avaliada como a média de todos os conjuntos de teste, sem levar em consideração o instante de tempo t.

O conjunto de teste não necessariamente precisa ocorrer de forma contígua imediatamente após a observação do conjunto de teste. A Figura 4.2 representa a avaliação da previsão de um passo a frente (*one-step-ahead forecast*), em contraste com a Figura 4.3 que avalia o desempenho de previsões em uma janela de quatro passos a frente (*4-step-ahead forecast*). Repare que no primeiro caso temos $h = 1$ e no segundo $h = 4$.

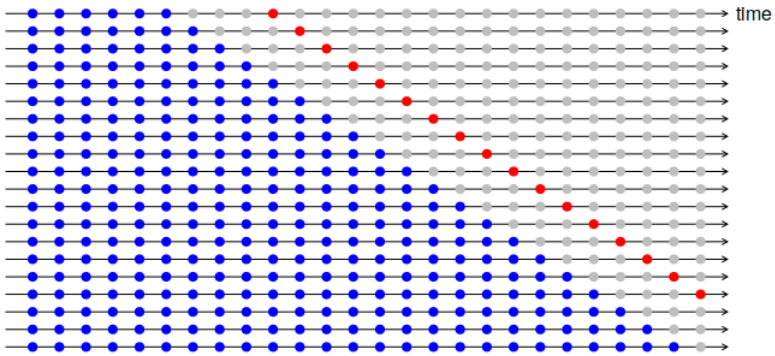


Figura 4.3: Avaliação deslizante com 4 passos a frente. Fonte [4].

Uma maneira comum de escolher o melhor modelo de previsão para uma série

temporal é escolher o modelo com o menor erro médio na validação cruzada, utilizando alguma métrica para o cálculo deste erro médio, como MAPE ou RMSE. No que segue, usaremos essas métricas de erro para avaliar e comparar diferentes modelos em diferentes conjuntos de séries temporais.

Capítulo 5

Conjunto de dados

As séries temporais utilizadas neste trabalho são provenientes de dados de fechamento de compras de clientes da VTEX no período de 01/01/2017 até 31/06/2019 (2 anos e 6 meses). Os dados brutos em JSON são guardados em um *data lake*, e campos relevantes destes dados são: data/hora da compra; *stock-keeping unit* (SKU), identificador (id) do produto comprado, id da categoria do produto, id da marca do produto, quantidade vendida do SKU na compra. O conjunto de dados bruto possui 14 terabytes, tendo em vista a grande quantidade de clientes e produtos vendidos neste período. O processamento deste enorme conjunto de dados a fim de construir séries temporais será descrito a seguir.

5.1 Construção das séries temporais

Figura 5.1: Pipeline do processamento de dados brutos para geração das séries temporais

Para a construção das séries temporais baseadas em produto, o motor de processamento distribuído Apache Spark [31] foi utilizado para realizar o ETL (Extract, Transform and Load [32]). A transformação utilizando Spark transforma os dados brutos em JSON para um conjunto de dados relacional em Parquet [33], no qual cada linha possui um vetor com as vendas observadas com a escala de tempo regular (por dia, semana ou mês) no intervalo definido, um *timestamp* de início da

```
{
  "accountName": "Conta exemplo",
  "productId": "32962",
  "productCategoryArray": "[662, 656]",
  "target": [
    580,
    342,
    971,
    132
  ],
  "start": "2017-01-01 00:00:00"
}
```

Listing 1: Exemplo de série temporal após processamento

série temporal e os identificadores de categorias (como id da marca, id do produto e id do departamento). O Exemplo 1 ilustra a série temporal de um produto com id 32962, pertencente as categorias 662 e 656 do varejista "Conta exemplo", com quatro meses (intervalo completo omitido por espaço) de vendas a partir das 0h do dia 01/01/2017, ou seja, o primeiro ponto representa as vendas entre 01/01/2017 e 01/02/2017; o segundo ponto entre 01/02/2017 e 01/03/2017 e assim por diante. O formato de dados Parquet foi utilizado por possuir diversas otimizações de armazenamento, principalmente para cargas de trabalho analíticas, além de ser o formato de arquivo padrão utilizado pelo Apache Spark [34].

O pipeline de transformação dos dados é resumido na Figura 5.1 e corresponde aos seguintes códigos pyspark preprocessing.py e ETL.py disponíveis no apêndice 7. Um outro conjunto de dados foi criado com informações exógenas de preços e vendas globais, porém - por questões computacionais - este conjunto não foi considerado para as análises dos modelos neste trabalho.

Para o processamento dos dados brutos um cluster com 408 vCPUs e 864 GiB de memória foram utilizados (25 C5 High-CPU Quadruple Extra Large e 1 R5 Double Extra Large [35]). O tempo total de execução do ETL foi de 16 horas e 21 minutos. Na Figura 5.2 podemos observar o custo associado ao processamento dos dados, no qual os saltos de custos diários estão relacionados a cada execução do processamento descrito, em torno de 700 dólares por execução do ETL.

Diferentes tipos de agregações foram realizadas: SKU, produto, categoria e conta, do menor nível de segmentação para o maior nível, respectivamente. Por exemplo, séries temporais de um tênis com tamanhos de 35 a 45, possui 11 SKUs. Uma agregação por produto soma as vendas de todos os 11 SKUs para cada instante de tempo t. Uma agregação por categoria de tênis de corrida, soma as vendas de

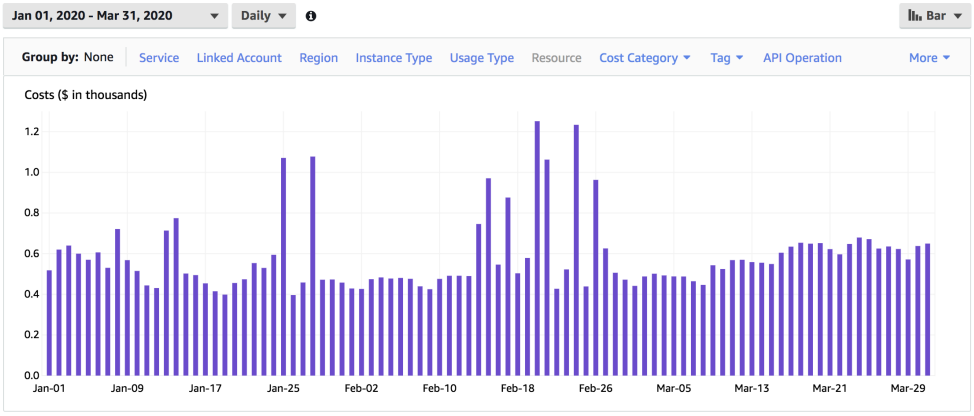


Figura 5.2: Custos diários de utilização de recursos

todos os produtos que possuem esta categoria. Uma agregação por conta soma as séries temporais de todas as categorias da loja, por exemplo: tênis de corrida, tênis casuais, camisas brancas, eletrônicos, perfumaria, *smartphones*, etc.

Segmento	Escala de tempo	Sazonalidade	Número de séries
Produto	1 dia	7 dias	11861
	7 dias	4 semanas	11861
	1 mês	12 meses	11861
Categoria	1 dia	7 dias	23898
	7 dias	4 semanas	23898
	1 mês	12 meses	23898
Conta	1 dia	7 dias	1633
	7 dias	4 semanas	1633
	1 mês	12 meses	1633

Tabela 5.1: Séries temporais obtidas dos dados processados

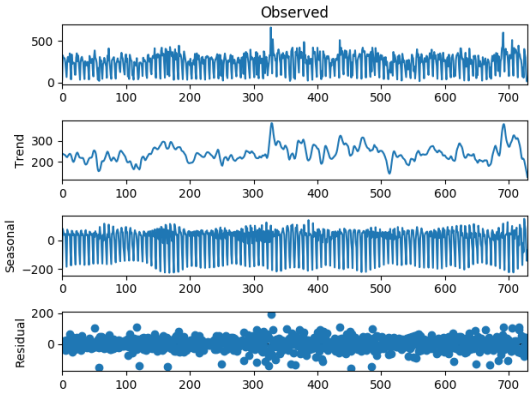
Para cada uma das segmentações supracitadas, séries temporais em diferentes escalas de tempo foram criadas: por dia, por semana e por mês. Deste modo, obtemos 9 conjuntos de dados, conforme resumido na Tabela 5.1.

Para cada série, novas séries temporais são criadas para a validação cruzada dos modelos: 14 lags para a escala de tempo de 1 dia; 8 lags para a escala de 7 dias e 7 lags para a escala de 1 mês. Repare que o número de séries temporais depende da agregação por segmento mas não da escala de tempo. Neste trabalho séries com alto volume de vendas foram escolhidas devido ao seu valor de negócio, o limiar definido esse é filtro foi de séries temporais com volume de vendas total maiores que 2200, compondo assim o conjunto de dados final. Outros agrupamentos poderiam ser formados de acordo com outros critérios, como características comuns entre as séries. Embora outros agrupamentos não tenham sido utilizados, veremos como o modelo DeepAR cria agrupamentos de forma automática na Seção 3.5 com

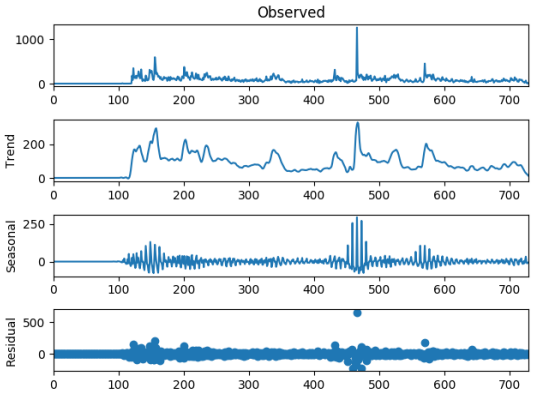
o objetivo de aumentar a acurácia do modelo.

5.2 Visualização das séries

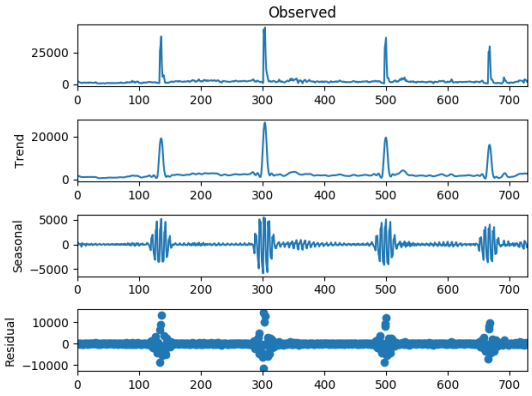
As figuras 5.3, 5.4 e 5.5 apresentam as séries temporais e as decomposições STL [36] para vendas de alguns produtos, categorias e contas, respectivamente, por dia. A Figura 5.6 representa a decomposição STL de vendas de produtos por mês. Decomposições STL são um método robusto e versátil para decomposição de séries temporais, lidando com qualquer tipo de sazonalidade e robusto a *outliers* [4]. Através dessa decomposição, podemos observar que os componentes sazonais e de inclinação não possuem padrões claros, nem mesmo no segmento de conta ou escala de tempo mensal.



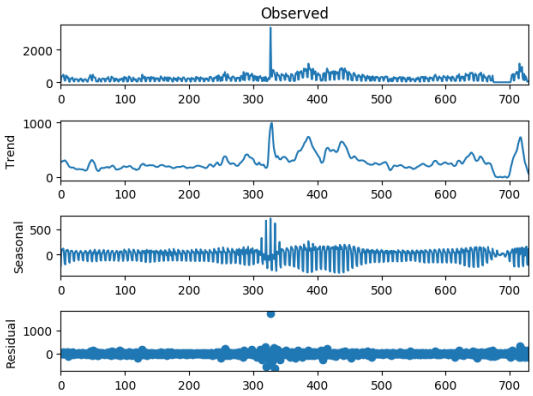
(a) Vendas da conta A por dia entre 2017 e 2018



(b) Vendas da conta B por dia entre 2017 e 2018



(c) Vendas da conta C por dia entre 2017 e 2018

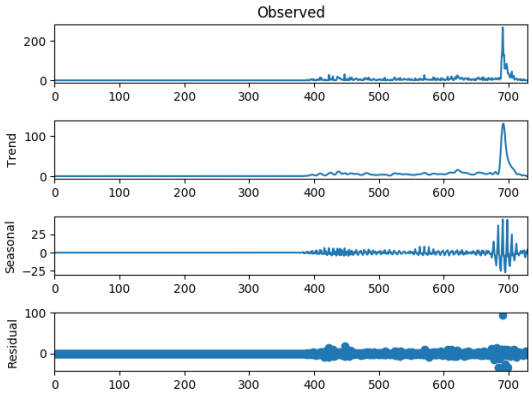


(d) Vendas da conta D por dia entre 2017 e 2018

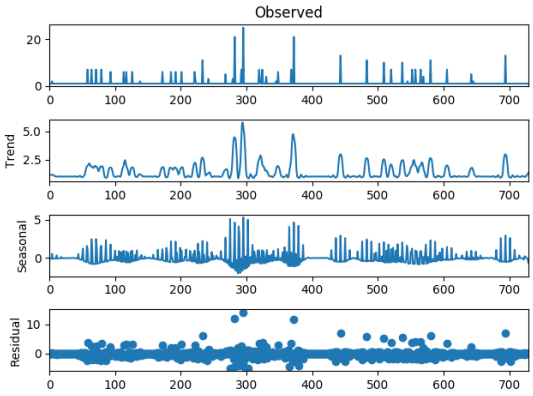
Figura 5.3: Decomposição das séries temporais de vendas de diferentes contas.

A Figura 5.3 apresenta a série temporal de quatro contas no período. Na Figura 5.3a podemos ver uma série com alto desvio padrão, nas Figuras 5.3b e 5.3c um comportamento chamado de *burst data*, um período de grande atividade de vendas

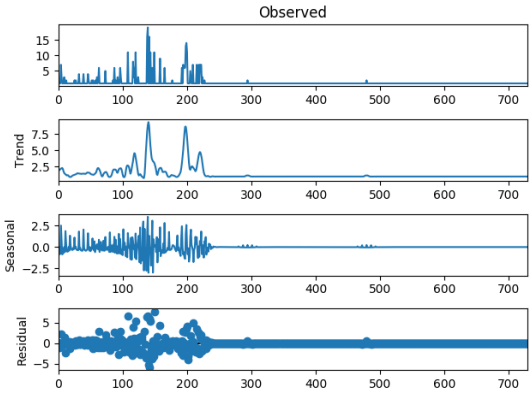
como promoções ou datas especiais (como a *Black Friday*). A Figura 5.3d além de conter *bursts* também existe um período anômalo sem vendas (próximo ao final da série), provavelmente devido a alguma alteração no código em produção que interrompeu o fluxo de vendas. As séries temporais de contas representam o maior nível de agregação da hierarquia, e estas figuras demonstram o comportamento errático até mesmo de lojas com alto volumes de vendas.



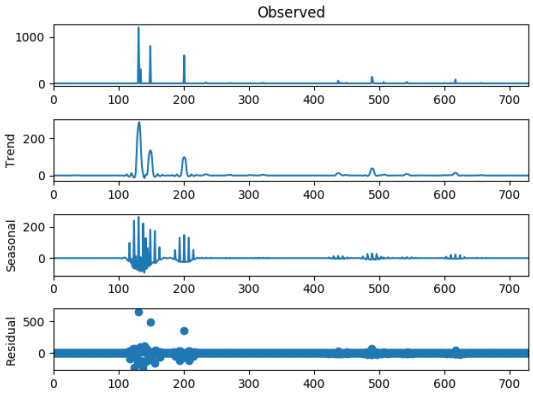
(a) Vendas para um produto da conta E



(b) Vendas para um produto da conta F



(c) Vendas para um produto da conta G

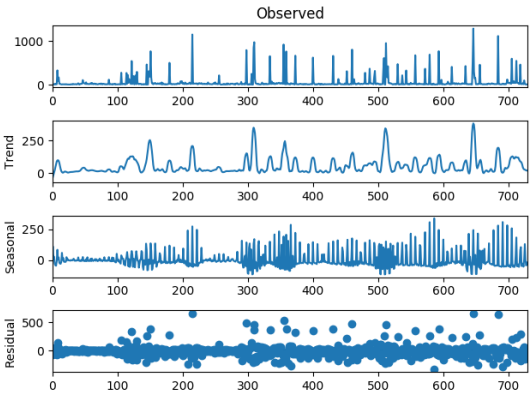


(d) Vendas para um produto da conta H

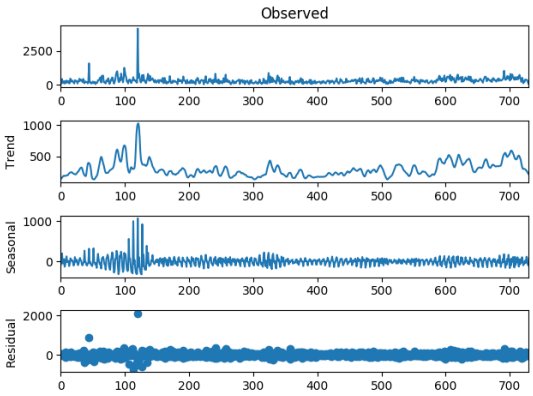
Figura 5.4: Decomposição das séries temporais de vendas de um produto de diferentes contas por dia entre 2017 e 2018.

A Figura 5.4 mostra a série temporal de quatro produtos de lojas diferentes. A Figura 5.4a mostra um produto que começou suas vendas no meio do período analisado; A Figura 5.4c um produto que interrompeu suas vendas (ou ficou sem estoque); A Figura 5.4d um produto com *burst* de vendas muito alto mas que no futuro apresentou pouca demanda; A Figura 5.4b demonstra um comportamento intermitente típico [37] das séries temporais deste conjunto de dados.

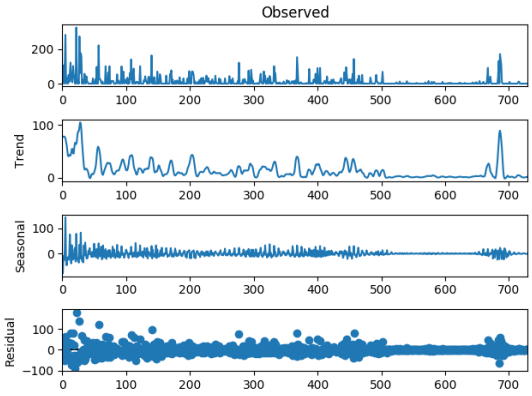
As séries temporais da Figura 5.5 representam quatro categorias de lojas diferentes e apresentam grande heteroscedasticidade [38], alternando em períodos de alta e baixa vendas pelas categorias, sendo a Figura 5.5d o maior exemplo deste comportamento. A Figura 5.6 mostra o nível de agregação por mês para produtos



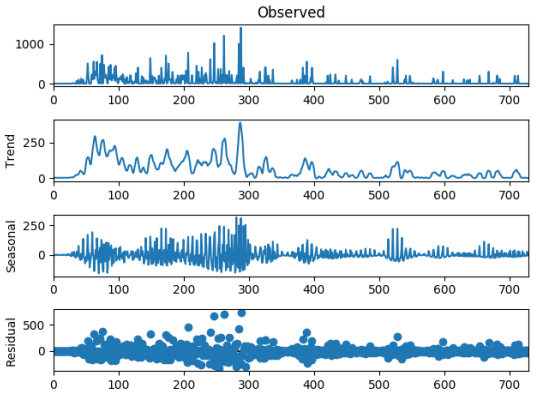
(a) Vendas para uma categoria da conta I



(b) Vendas para uma categoria da conta J



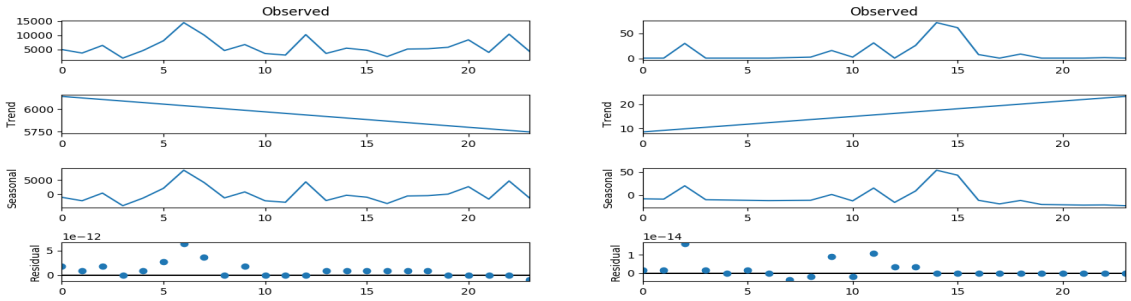
(c) Vendas para uma categoria da conta K



(d) Vendas para uma categoria da conta L

Figura 5.5: Decomposição das séries temporais de vendas de uma categoria de diferentes contas por dia entre 2017 e 2018.

de duas lojas. Embora os valores observados estejam suavizados por conta da escala de tempo mensal, os padrões das séries continuam não sendo muito claros.



(a) Vendas para um produto da conta M

(b) Vendas para um produto da conta N

Figura 5.6: Decomposição das séries temporais de vendas de um produto de diferentes contas por mês entre 2017 e 2018.

5.3 Estatísticas das séries

As Tabelas 5.2, 5.3 e 5.4 contém a estatística descritiva das médias de diferentes estatísticas das séries analisadas (a média do desvio padrão calculada para cada série, por exemplo). Embora o período de tempo de cada série temporal tenha sido fixado de antemão (de 01/01/2017 até 31/06/2019), pequenas variações entre os segmentos de tempo existem devido a modificações do início ou fim de uma série para completar sua respectiva sazonalidade, ou seja, estatísticas como médias das somas diferem de um pequeno montante.

Alguns aspectos gerais importantes comum a todas as estatísticas descritivas de segmentos: como esperado, as médias das estatísticas crescem de produto, para categorias até contas. A relação da média e mediana com o máximo diminui com a escala de tempo em todos os segmentos. A relação entre a média do mínimo e a média do máximo diminui quanto maior o nível de agregação. O mesmo acontece para a relação entre a média da mediana e a média do máximo. Outro aspecto geral importante é a média maior que mediana em todas as estatísticas, o que caracteriza cauda pesada das distribuições que pode ser observada nas figuras (com escala logarítmica) 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.13, 5.14 e 5.15.

Para a Tabela 5.2, podemos notar que algumas séries possuem médias dos mínimos bem próximos de zero para o intervalo de tempo diário, caracterizando séries intermitentes. Tabela 5.3 também possui o mesmo comportamento, em menor grau, com média do mínimo menor que 1.

Nas Figuras 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.13, 5.14 e 5.15 temos as funções de distribuição acumulada empírica complementares (*empirical complementary cumulative distribution function*, e.c.c.d.f)) da soma, desvio padrão e média das séries

	Média		
Estatística	1 dia	7 dias	1 mês
Soma	4218.63	4239.69	4167.96
Média	5.72	40.00	173.67
Mediana	2.24	19.53	96.27
Desvio Padrão	13.43	68.40	224.62
Mínimo	0.01	1.59	19.44
Máximo	178.37	458.93	915.28

Tabela 5.2: Estatística descritiva para o segmento de produtos

	Média		
Estatística	1 dia	7 dias	1 mês
Soma	17788.01	17889.61	17555.68
Média	24.14	168.77	731.49
Mediana	14.14	109.55	515.53
Desvio Padrão	36.54	189.47	641.74
Mínimo	0.86	23.52	175.10
Máximo	459.47	1219.74	2624.56

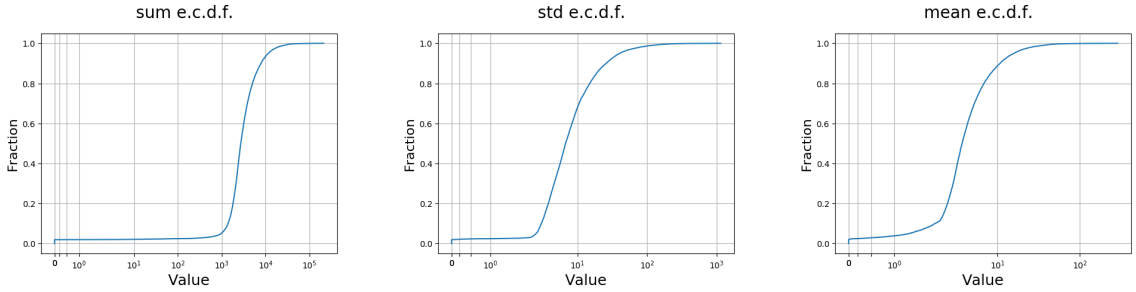
Tabela 5.3: Estatística descritiva para o segmento de categorias

	Média		
Estatística	1 dia	7 dias	1 mês
Soma	101232.42	101804.88	99975.82
Média	137.36	960.42	4165.66
Mediana	95.20	708.36	3264.77
Desvio Padrão	169.47	875.85	2944.06
Mínimo	10.93	207.39	1399.81
Máximo	2084.56	5583.71	12491.16

Tabela 5.4: Estatística descritiva para o segmento de contas

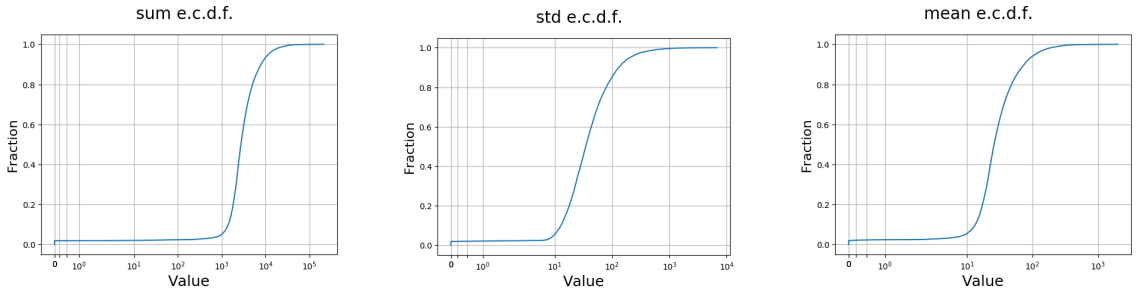
temporais em diferentes segmentos e escala de tempo. Por exemplo, a Figura 5.9 mostra a e.c.c.d.f da média das séries temporais de produtos na escala de tempo mensal. Repare que o eixo-x está em escala logarítmica. Podemos ver que quanto maior o nível de agregação, maior a disparidade entre as séries em todos os gráficos. Por exemplo, na Figura 5.15 representando contas mensais, temos que o volume total de vendas para a maioria das séries gira em torno de 10^3 até 10^6, porém ainda existe uma pequena proporção entre 0 e 10^3 e 10^6 até 10^7. Da mesma forma existe uma grande variedade de desvios padrões, com concentração entre 10^1 até 10^4, como ilustrado na Figura 5.15.

Olhando pelo lado das séries com menos volume, na Figura 5.9 representando produtos mensais, podemos verificar que existe uma grande concentração na soma entre 10^3 até 10^4 e desvio padrão similar ao por contas mensais. De acordo com a Tabela 5.2, podemos observar nas figuras 5.9, 5.8 e 5.7 que o desvio padrão aumenta de acordo com a escala de tempo, além de possuir maior variação nas séries temporais por dia. Esses padrões se repetem também para categorias e contas.



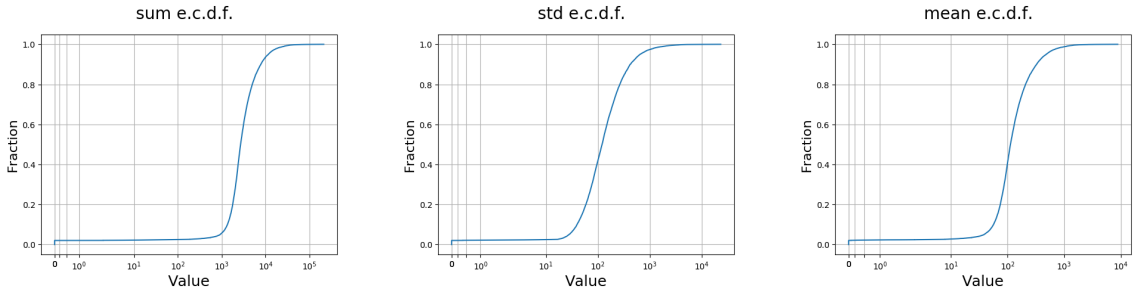
(a) Soma (b) Desvio Padrão (c) Média

Figura 5.7: e.c.c.d.f's para séries temporais de produtos por dia.



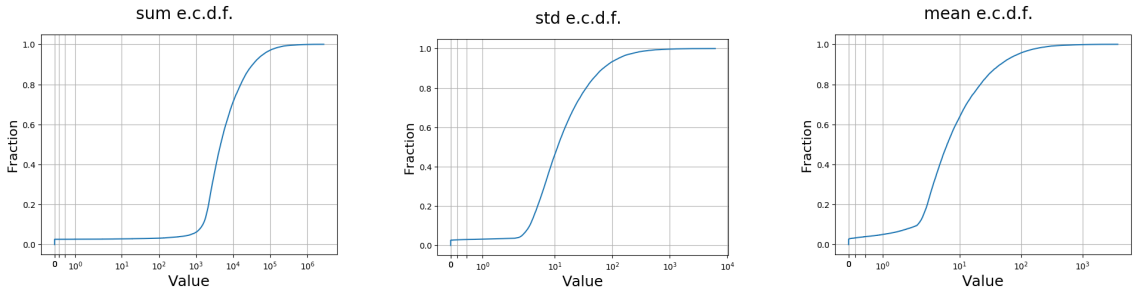
(a) Soma (b) Desvio Padrão (c) Média

Figura 5.8: e.c.c.d.f's para séries temporais de produtos por 7 dias.



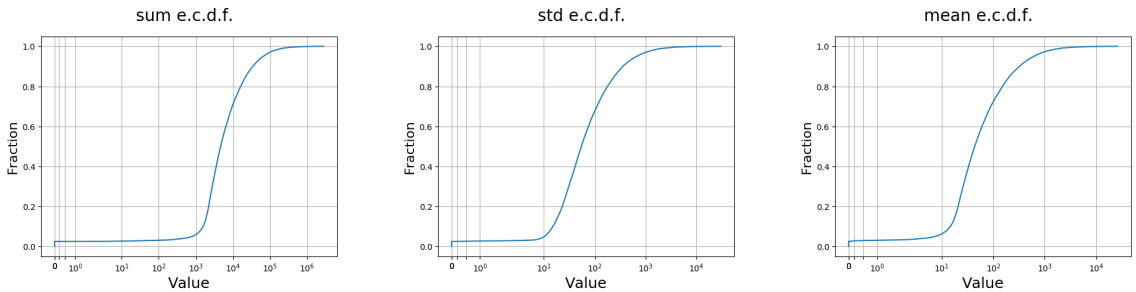
(a) Soma (b) Desvio Padrão (c) Média

Figura 5.9: e.c.c.d.f's para séries temporais de produtos por mês.



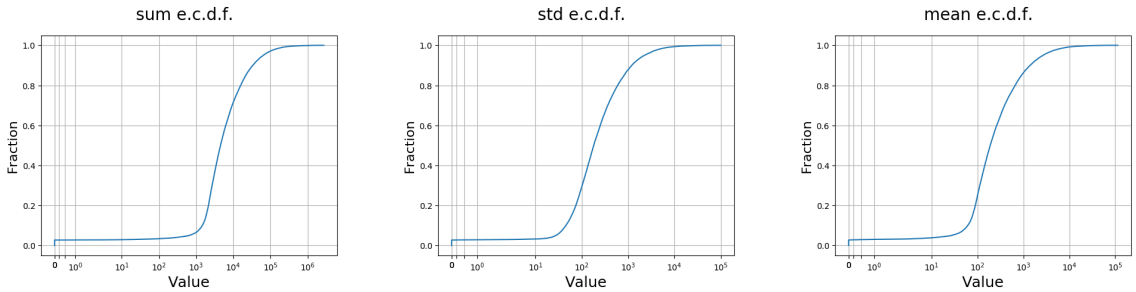
(a) Soma (b) Desvio Padrão (c) Média

Figura 5.10: e.c.c.d.f's para séries temporais de categorias por dia.



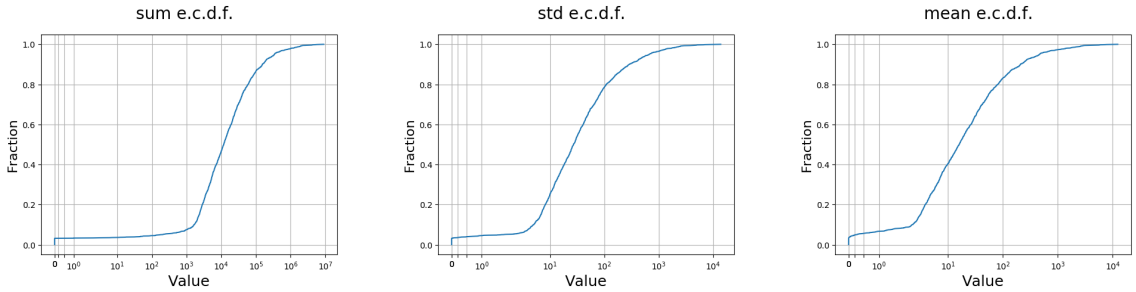
(a) Soma (b) Desvio Padrão (c) Média

Figura 5.11: e.c.c.d.f's para séries temporais de categorias por 7 dias



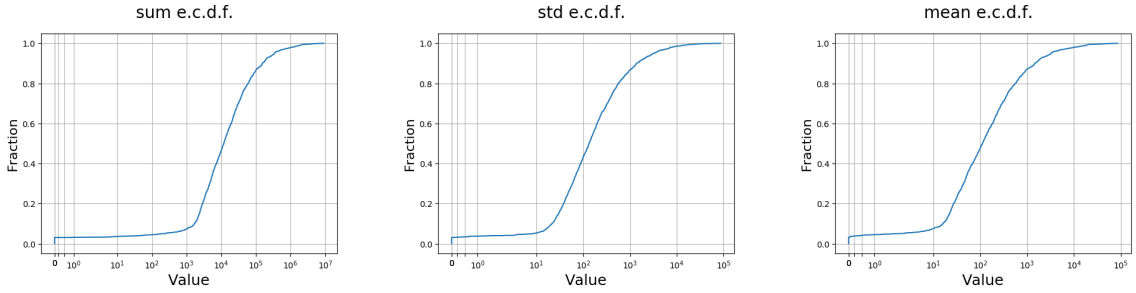
(a) Soma (b) Desvio Padrão (c) Média

Figura 5.12: e.c.c.d.f's para séries temporais de categorias por mês.



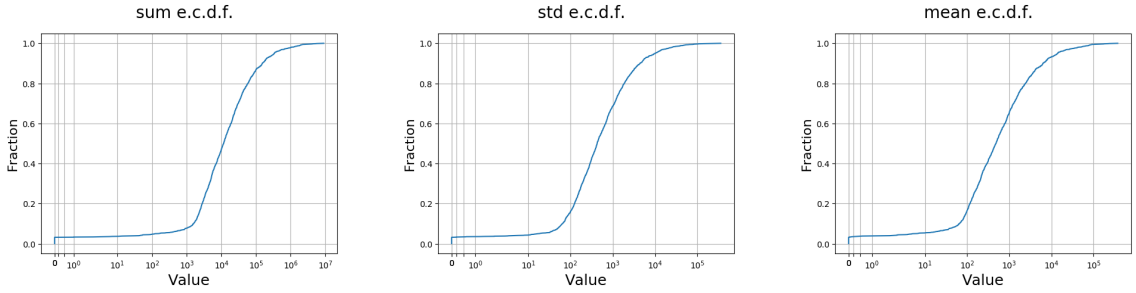
(a) Soma (b) Desvio Padrão (c) Média

Figura 5.13: e.c.c.d.f's para séries temporais de contas por dia.



(a) Soma (b) Desvio Padrão (c) Média

Figura 5.14: e.c.c.d.f's para séries temporais de contas por 7 dias.



(a) Soma (b) Desvio Padrão (c) Média

Figura 5.15: e.c.c.d.f's para séries temporais de contas por mês.

Capítulo 6

Avaliação dos modelos

Os modelos serão avaliados utilizando 261 conjuntos de dados independentes que foram criados a partir das séries temporais para a validação cruzada. A criação dos 261 conjuntos foram a partir dos cortes das séries temporais originais. Se uma série temporal possui tamanho N, podemos criar k previsões independentes para avaliar um modelo truncando a série temporal original em k séries temporais de tamanho $N-0$, $N-1$, ..., $N-k-1$, mediante a remoção de $i \in \{0,\ldots,k-1\}$ pontos a partir do último ponto observado. Para uma avaliação cruzada com t passos a frente, para cada um dos truncamentos $N-i$ com $i \in \{0,\ldots,k-1\}$ realizamos a previsão de $N-i+t$. Neste trabalho, para o intervalo diário foram utilizados 14 previsões ($k=14$), 8 previsões para semanas ($k=8$) e 7 previsões ($k=7$) para meses. Totalizando $(14+8+7)*3*3 = 261$ conjuntos, com o primeiro 3 da equação devido ao número de segmentos (produto, categoria e conta) e o segundo 3 devido a cada uma das escalas de tempo (diária, semanal e mensal). Cada um dos conjuntos de dados foram criados com dois passos a frente (Seção 4.4) e a métrica de erro do primeiro passo foi descartada. Logo, uma previsão mensal receberia como entrada uma série temporal até dezembro e realizaria a previsão de janeiro e fevereiro. A previsão de janeiro seria descartada e o erro é calculado com a previsão de fevereiro e o verdadeiro valor de fevereiro. O motivo da predição de dois passos a frente é para incorporar na modelagem o tempo de reação do tomador de decisão, deixando o primeiro passo para isto. Modelar o tempo de reação do tomador de decisão torna o problema mais difícil que prever apenas um passo a frente (Figura 6.2), no qual a proximidade temporal torna técnicas simples viáveis já que pontos adjacentes possuem alta correlação. As métricas utilizadas para os relatórios foram MAE e MAPE, devido ao fácil entendimento do tomador de decisão com estas métricas. As distribuições de erros foram geradas e o melhor modelo foi escolhido de acordo com o menor erro médio.

Para a geração das tabelas e figuras deste capítulo uma máquina virtual M5 General Purpose 16xlarge (256.0 GiB, 64 vCPUs) da AWS foi utilizada devido ao

tamanho dos arquivos de saída dos modelos que possuem mais de 170GB (Figura 6.1).

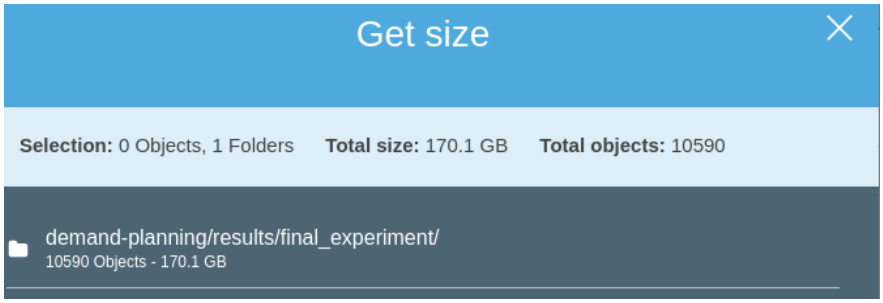


Figura 6.1: Arquivos de saída com os resultados do modelo

A pré-seleção das séries temporais (volume total maior que 2200), aumentou a dificuldade das previsões, já que séries temporais com pouca demanda são fáceis de prever (a previsão trivial de 0 para todos os pontos obteria alta acurácia). Este comportamento pode ser observado em um experimento inicial com uma restrição menor das séries temporais analisadas. Na Figura 6.2 podemos observar que em todos os modelos avaliados, 80% das séries temporais possuem erro MAPE menor que 0.5 para 1.220.039 produtos, um desempenho significativamente melhor do que será apresentado neste trabalho. Porém, a previsão destas séries temporais tem pouco valor de negócio, devido ao grande volume de vendas dos produtos mais vendidos, que são mascarados neste conjunto de séries de baixa demanda (Figura 1.1).

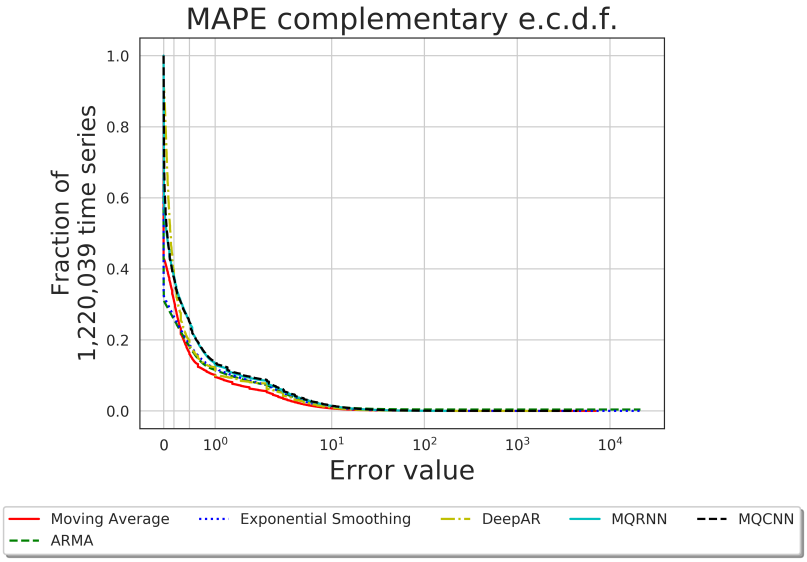


Figura 6.2: Comparando os modelos com um critério de pré-seleção menos restrito que o utilizado no trabalho para as séries temporais de produtos.

6.1 Metodologia

O fluxograma na Figura 6.3 descreve a metodologia utilizada neste trabalho, em suma:

1. Os dados são processados de acordo com o Capítulo 5;
2. Cada uma das séries temporais pertencentes aos dados da etapa 1 são transformados de acordo com a Subseção 6.1.1;
    - Para cada série temporal - original e transformada - é realizada a decomposição STL e estatísticas de acordo com a Seção 5.2.
3. É realizada a validação cruzada descrita na Seção 4.4;
    - Os resultados de todos os modelos são registrados para a escolha do melhor modelo e também para a estratégia "Best"(Subseção 6.1.4).
4. São gerados gráficos da e.c.c.d.f. (*empirical complementary cumulative distribution function*) dos erros; gráficos de dispersão dos erros (MAPE ou MAE) vs. estatísticas descritivas (como média, soma e desvio padrão) das séries e por fim gráficos da distribuição dos erros completa de cada série de um modelo para análise.

A metodologia será descrita mais detalhadamente nas próximas subseções.

6.1.1 Transformação das séries

Técnicas de pré-processamento como a transformação de Box-Cox [39], detrend [40] e diferenciação foram utilizadas, mas não obtiveram melhorias estatisticamente significativas. Isto pode ser devido ao fato da criação de modelos automáticos para um grande número de séries temporais: séries que obtiveram piores resultados mascaram as séries que possuem um melhor desempenho com as técnicas aplicadas. Uma forma de resolver este problema seria descobrir as melhores técnicas de pré-processamento para cada série temporal (ou agrupamento de séries), mas um custo computacional ainda maior seria necessário para esta análise.

6.1.2 Modelos

Os modelos de previsão utilizados foram de média móvel (MA) (Seção 2.4.2, também conhecido como *Rolling Average* ou *Running Average*), no qual MA(n) refere-se a utilização dos últimos n pontos para o cálculo da média; Suavizações exponenciais (Seção 2.3) com seus parâmetros otimizados para cada série temporal; ARMA(0,1)

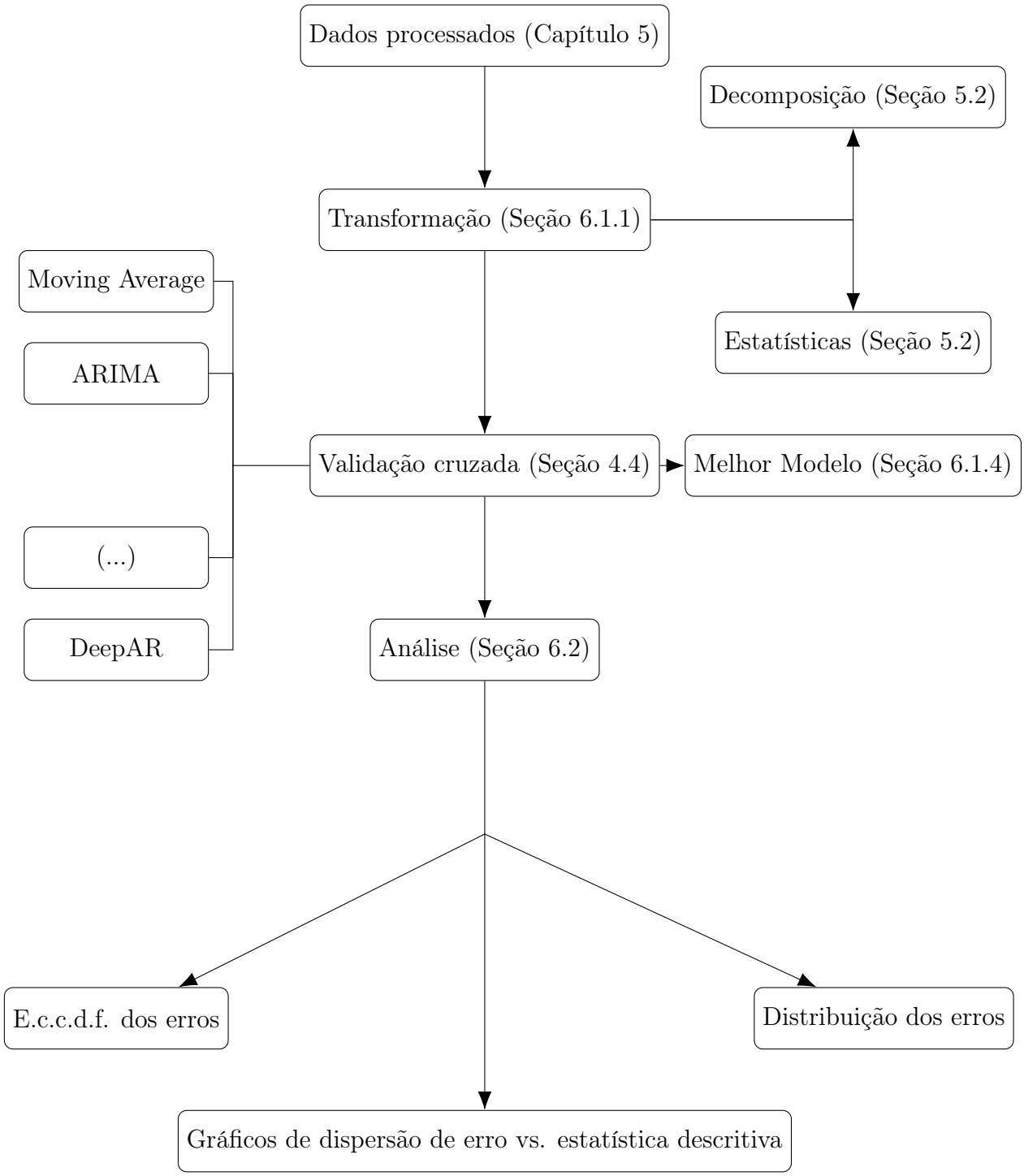


Figura 6.3: Metodologia

(Seção 2.4.4); Prophet [41] e DeepAR (Seção 3.5) com 40 neurônios, duas camadas e 100 amostras para cada previsão do modelo. A janela de contexto é de 7 dias para intervalo diário; 4 semanas para intervalo semanal e 1 mês para o intervalo mensal. A janela de predição foi sempre de dois pontos consecutivos. DeepAR(q) refere-se a utilização do percentil q do traço do modelo. A Tabela 5.1 foi utilizada para os modelos DeepAR e Prophet, que aceitam como entrada a sazonalidade.

Os modelos ETS [6] da biblioteca *forecast* [42] e modelos hierárquicos (ex.: [43]) da linguagem R não foram utilizados devidos a limitações técnicas de paralelização, no qual somente um modelo pode ser treinado de cada vez, o que impossibilita o treinamento em grandes volumes de dados e grande quantidade de séries temporais. Atributos exógenos foram criados a partir dos dados brutos para melhorar a previsão, mas modelos clássicos com variáveis exógenas mostraram-se impraticáveis computacionalmente, tornando seu uso inviável. Dessa forma, as variáveis exógenas não foram utilizadas nos métodos com redes neurais permitindo assim uma melhor comparação entre modelos. O método ARIMA, MQ-RNN [44] e MQ-CNN [44] foram descartados da análise por não convergirem para a maioria das séries, sendo utilizado somente o modelo ARMA$(0, 1)$ do método ARIMA. O modelo DeepFactor [9] obteve desempenho muito inferior em séries com baixas vendas e também foi excluído das análises.

6.1.3 Treinamento local e global

Todos os modelos avaliados exceto o modelo DeepAR, são modelos locais, ou seja, diferentes parâmetros são ajustados para cada série temporal. Isso significa $\underbrace{(14+8+7)}_{\text{número de previsões}} * \underbrace{3}_{\text{produtos, categorias e contas}} * \underbrace{11861}_{\text{séries de produtos}} = 1031907$ modelos para produtos, $(14 + 8 + 7) * 3 * \underbrace{23898}_{\text{séries de categorias}} = 2079126$ para categorias e $(14 + 8 + 7) * 3 * \underbrace{1633}_{\text{séries de contas}} = 142071$ para contas. Em comparação, métodos globais como o DeepAR compartilham os pesos ajustados que são treinados sobre todas as séries temporais. Isso significa um ganho em escalabilidade para grandes conjuntos de dados. Abordagens mistas como o DeepFactor [9] também são possíveis, embora neste trabalho os resultados não se demonstraram satisfatórios. Neste trabalho foram utilizados 7 modelos locais e 3 modelos globais, isto significa um total de $\underbrace{7}_{\text{número de modelos locais}} * \overbrace{(\underbrace{1031907}_{\text{produtos}} + \underbrace{2079126}_{\text{categorias}} + \underbrace{142071}_{\text{contas}})}^{\text{modelos}} + \underbrace{7}_{\text{número de modelos globais}} * \underbrace{3}_{\text{segmentos}} * \underbrace{3}_{\text{escalas de tempo}} = 22771728 + 27 = 22771755$ modelos treinados. O tempo total de treino dos modelos foi de 5 dias em uma máquina virtual C5 High-CPU 18xlarge

da AWS [35] com 144.0 GiB de memória e 72 vCPUs (vCPU é uma *thread* de um Intel Xeon core). Todo esse processo de treinamento precisa ser orquestrado para serem executados periodicamente, previsões diárias precisam ser realizadas com no mínimo um dia de antecedência; previsões semanais com no mínimo com uma semana de antecedência e mensais com no mínimo um mês de antecedência. Modelos como média móvel não necessitam de etapa de retreinamento quando estão em produção, sempre consumindo novos dados e atualizando de maneira online seus parâmetros.

6.1.4 Utilização mista de melhores modelos

Raramente um determinado modelo terá o melhor desempenho médio para *todas* as séries temporais de um conjunto de dados. Neste trabalho, o melhor desempenho final é definido como o mínimo entre as médias dos modelos calculadas na validação cruzada. Por exemplo, para um conjunto de dados com duas séries temporais, se dois modelos obtivessem 1.0 e 2.0 para a métrica MAPE na primeira série temporal e 4.0 e 3.0 para a métrica MAPE na segunda série temporal, o melhor desempenho seria de $\frac{(1.0+3.0)}{2} = 2.0$ com a escolha do modelo 1 para a primeira série e do modelo 2 para a segunda série. Como não houve treinamento de hiper-parâmetros neste trabalho (apenas de parâmetros) por questões computacionais, a escolha dos melhores modelos para cada série temporal pode ser feito diretamente, já que calculamos o erro médio através da validação cruzada para os respectivos truncamentos de cada série.

Os parâmetros de cada modelo treinado é armazenado e utilizado posteriormente para realizar a previsão.

6.2 Avaliação

As Tabelas 6.2, 6.3 e 6.4 apresentam o erro médio de cada modelo para as três escalas de tempo e os três segmentos, respectivamente. Nota-se que em praticamente todos os modelos o intervalo de tempo diário possui a menor média de erros, seguido pelo intervalo mensal e finalmente semanal. A estratégia de melhor valor médio prova-se útil obtendo melhorias significativas em relação ao uso de somente um modelo para todas as previsões das séries temporais (indicado na última linha). Na Tabela 6.2 temos que a estratégia de melhor valor no segmento de produtos foi 26.2%, 60% e 43% melhor que o melhor modelo (DeepAR(0.4)) na métrica MAPE para a escala de tempo de 1 dia, 7 dias e 1 mês, respectivamente. Destaques para as escalas de tempo de 7 dias e 1 mês, com ganhos percentuais médios de 45.3% e 63.6%, respectivamente. Os resultados de ganhos percentuais para todos segmentos

	Produtos			Categorias			Contas		
	1 dia	7 dias	1 mês	1 dia	7 dias	1 mês	1 dia	7 dias	1 mês
Melhor modelo	0.77	1.07	1.00	0.78	0.83	0.87	0.74	0.77	1.21
Melhor valor	0.61	0.67	0.70	0.62	0.58	0.52	0.61	0.58	0.67
Ganho percentual	26.2%	60%	43%	25.9%	43.1%	67.3%	21.3%	32.8%	80.6%

Tabela 6.1: Ganho percentual da estratégia de melhor valor em relação ao melhor modelo de cada segmento e escala de tempo. O melhor modelo foi DeepAR(0.4) para todos os segmentos e escalas de tempo

e escalas de tempo podem ser analisados na Tabela 6.1.

Como comportamentos gerais dos erros, podemos examinar através das tabelas de resultados (6.2, 6.3 e 6.4) que os modelos têm desempenho médio (relativamente) parecidos e que um modelo muito simples como MA tem bom desempenho e se mostram superiores em comparação a modelos sofisticados como o DeepAR para alguns percentis ou Prophet. Isso demonstra a importância da etapa de seleção de modelos antes do comprometimento ao ajuste de modelos mais sofisticados, visto que um modelo simples que pode até mesmo realizar previsões de maneira *online* tem um bom desempenho. Outro ponto geral que podemos observar é que os erros também tendem a aumentar com a escala de tempo. Na métrica MAE isso é esperado, dado que os valores são tratados em termos absolutos, mas este comportamento também ocorre na métrica MAPE que é invariante a escala. Isto vai de acordo com a intuição da maior dificuldade e incerteza associada na previsão de horizontes de tempo mais longos.

Temos que os melhores modelos em ordem ascendente no segmento de produtos (Tabela 6.5), para o intervalo diário, são: DeepAR(0.4), DeepAR(0.5) e ARMA (Tabela 6.2). No segmento de categorias (Tabela 6.3): DeepAR(0.4), DeepAR(0.5) e MA(7). No segmento de contas DeepAR(0.4), DeepAR(0.5) e MA(3). O modelo DeepAR obteve o melhor desempenho no intervalo diário, seguido por MA(7) (com a exceção do modelo ARMA$(0, 1)$ no segmento de produtos).

Para o intervalo semanal no segmento de produtos, temos como melhores em ordem ascendente (Tabela 6.6): DeepAR(0.4), ETS(N,N) e DeepAR(0.5) (Tabela 6.2); Para o segmento de categorias: DeepAR(0.4), DeepAR(0.5) e ETS(N,N) (Tabela 6.3). Para o segmento de contas: DeepAR(0.4), DeepAR(0.5) e ETS(N,N) (Tabela 6.4). Novamente o modelo DeepAR obteve o melhor desempenho geral, seguido por ETS(N,N).

Para o intervalo mensal no segmento de produtos, temos como melhores (Tabela 6.7): DeepAR(0.4), DeepAR(0.5) e DeepAR(0.6) (Tabela 6.2). Para o segmento de categorias: DeepAR(0.4), DeepAR(0.5) e ETS(N,N) (Tabela 6.3). Para o segmento de contas: ETS(N,N), DeepAR(0.5) e DeepAR(0.6) (Tabela 6.4). Novamente com o

DeepAR se mostra superior, seguido pela média móvel MA(3). Isto demonstra que no caso mensal que possuem poucos pontos (30 meses no total), o modelo DeepAR não realiza um sobreajuste, sendo ainda assim superior aos modelos simples.

Desta forma, o melhor modelo para o conjunto de dados do estudo foi o modelo DeepAR, seguido por uma simples média móvel. Podemos notar que embora o modelo DeepAR tenha obtido o melhor desempenho geral, a escolha natural de utilizar a mediana da distribuição do traço não obteve o melhor desempenho (DeepAR(0.5)). Isto demonstra a importância da utilização de diferentes percentis para alguns conjuntos. Também podemos observar a estabilidade entre os percentis utilizados: com o desempenho do percentil 0.4, 0.5, 0.6 obtendo os melhores resultados, nessa ordem, para todos os experimentos.

Produto						
Modelo	MAPE			MAE		
	1 dia	7 dias	1 mês	1 dia	7 dias	1 mês
MA(3)	1.09	1.54	1.63	4.78	51.58	137.94
MA(7)	1.01	1.52	1.63	4.41	50.34	152.14
ETS(A,A)	1.25	1.59	2.79	5.63	69.56	270.10
ETS(A_d,A)	1.23	1.48	2.59	5.59	67.15	260.50
ETS(N,N)	1.07	1.35	1.82	4.62	52.56	142.45
ARMA	0.99	1.53	2.02	8.10	71.45	234.17
Prophet	1.47	1.42	2.33	7.63	58.17	182.05
DeepAR(0.4)	0.77	1.07	1.00	4.20	46.09	134.24
DeepAR(0.5)	0.88	1.36	1.24	4.22	47.54	135.59
DeepAR(0.6)	1.09	1.68	1.59	4.54	51.65	153.94
Melhor valor	0.61	0.67	0.70	3.60	38.33	94.25

Tabela 6.2: Resultados dos modelos para o segmento de produtos

Categoria						
Modelo	MAPE			MAE		
	1 dia	7 dias	1 mês	1 dia	7 dias	1 mês
MA(3)	1.06	1.37	1.25	17.99	179.29	395.16
MA(7)	1.02	1.26	1.33	15.99	170.34	442.36
ETS(A,A)	1.27	1.31	2.61	21.56	230.09	747.47
ETS(A_d,A)	∞	1.23	2.54	∞	224.6	724.23
ETS(N,N)	1.07	1.16	1.09	17.32	181.34	383.72
ARMA	2.34	3.8	4.84	41.14	356.92	1187.96
Prophet	1.54	1.30	1.75	23.79	184.67	493.91
DeepAR(0.4)	0.78	0.83	0.87	15.81	156.61	484.84
DeepAR(0.5)	0.91	1.14	0.96	16.02	164.48	452.38
DeepAR(0.6)	1.10	1.36	1.23	16.92	182.69	404.02
Melhor valor	0.62	0.58	0.52	13.40	137.17	296.43

Tabela 6.3: Resultados dos modelos para o segmento de categorias

Conta						
Modelo	MAPE			MAE		
	1 dia	7 dias	1 mês	1 dia	7 dias	1 mês
MA(3)	1.00	1.42	1.66	91.00	864.17	1764.37
MA(7)	1.00	1.16	2.29	77.82	814.38	1996.50
ETS(A,A)	1.25	1.18	4.96	111.10	993.74	3307.90
ETS(A_d,A)	1.22	1.12	4.71	107.84	961.94	3191.60
ETS(N,N)	1.03	1.07	2.05	87.01	836.29	1700.65
ARMA	7.50	11.61	15.57	273.47	2247.44	7612.73
Prophet	1.85	1.17	2.18	107.35	846.71	2220.95
DeepAR(0.4)	0.74	0.77	1.21	71.08	763.72	1959.57
DeepAR(0.5)	0.90	0.87	1.38	75.71	759.13	1757.95
DeepAR(0.6)	1.08	1.35	1.67	75.79	884.56	1763.94
Melhor valor	0.61	0.58	0.67	63.78	653.37	1347.20

Tabela 6.4: Resultados dos modelos para o segmento de contas

	Segmento		
	Produtos	Categorias	Contas
1º	Best	Best	Best
2º	DeepAR(0.4)	DeepAR(0.4)	DeepAR(0.4)
3º	DeepAR(0.5)	DeepAR(0.5)	DeepAR(0.5)
4º	ARMA	MA(7)	MA(3)
5º	MA(7)	MA(3)	MA(7)
6º	ETS(N,N)	ETS(N,N)	ETS(N,N)
7º	MA(3)	DeepAR(0.6)	DeepAR(0.6)
8º	DeepAR(0.6)	ETS(A,A)	ETS(A_d,A)
9º	ETS(A_d,A)	Prophet	ETS(A,A)
10º	ETS(A,A)	ARMA	Prophet
11º	Prophet	ETS(A_d,A)	ARMA

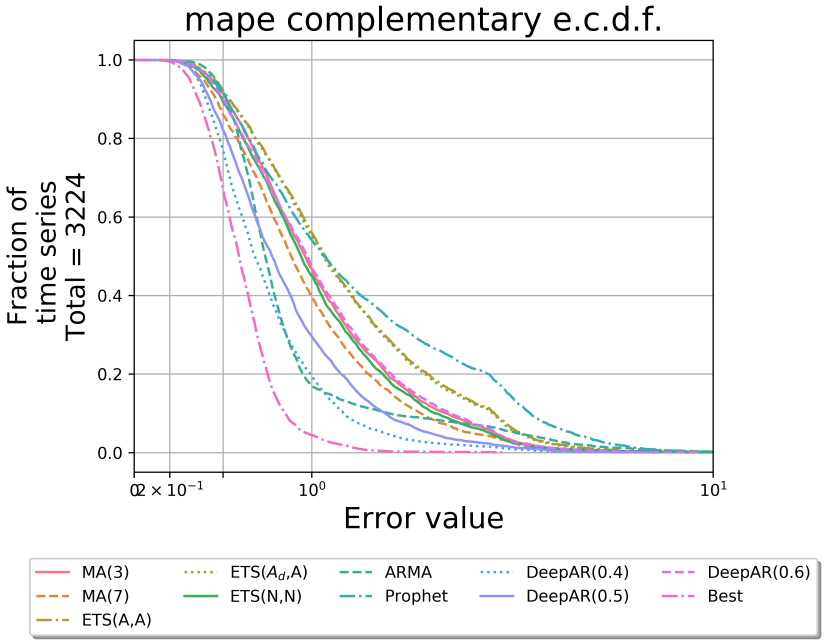
Tabela 6.5: *Ranking* de modelos para o intervalo diário

	Segmento		
	Produtos	Categorias	Contas
1º	Best	Best	Best
2º	DeepAR(0.4)	DeepAR(0.4)	DeepAR(0.4)
3º	ETS(N,N)	DeepAR(0.5)	DeepAR(0.5)
4º	DeepAR(0.5)	ETS(N,N)	ETS(N,N)
5º	Prophet	ETS(A_d,A)	ETS(A_d,A)
6º	ETS(A_d,A)	MA(7)	MA(7)
7º	MA(7)	Prophet	Prophet
8º	ARMA	ETS(A,A)	ETS(A,A)
9º	MA(3)	DeepAR(0.6)	DeepAR(0.6)
10º	ETS(A,A)	MA(3)	MA(3)
11º	DeepAR(0.6)	ARMA	ARMA

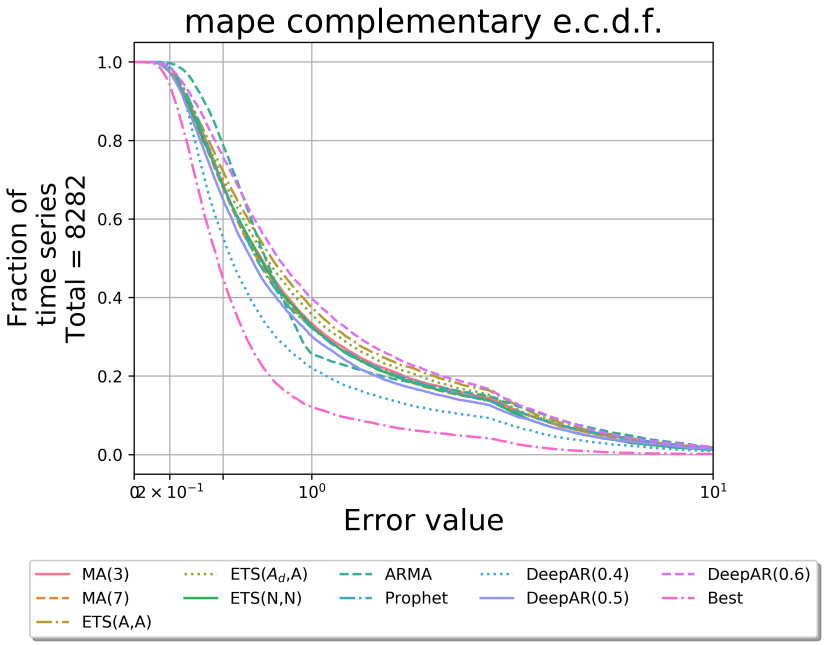
Tabela 6.6: *Ranking* de modelos para o intervalo semanal

	Segmento		
	Produtos	Categorias	Contas
1º	Best	Best	Best
2º	DeepAR(0.4)	DeepAR(0.4)	DeepAR(0.4)
3º	DeepAR(0.5)	DeepAR(0.5)	DeepAR(0.5)
4º	DeepAR(0.6)	ETS(N,N)	MA(3)
5º	MA(3)	DeepAR(0.6)	DeepAR(0.6)
6º	MA(7)	MA(3)	ETS(N,N)
7º	ETS(N,N)	MA(7)	Prophet
8º	ARMA	Prophet	MA(7)
9º	Prophet	ETS(A_d,A)	ETS(A_d,A)
10º	ETS(A_d,A)	ETS(A,A)	ETS(A,A)
11º	ETS(A,A)	ARMA	ARMA

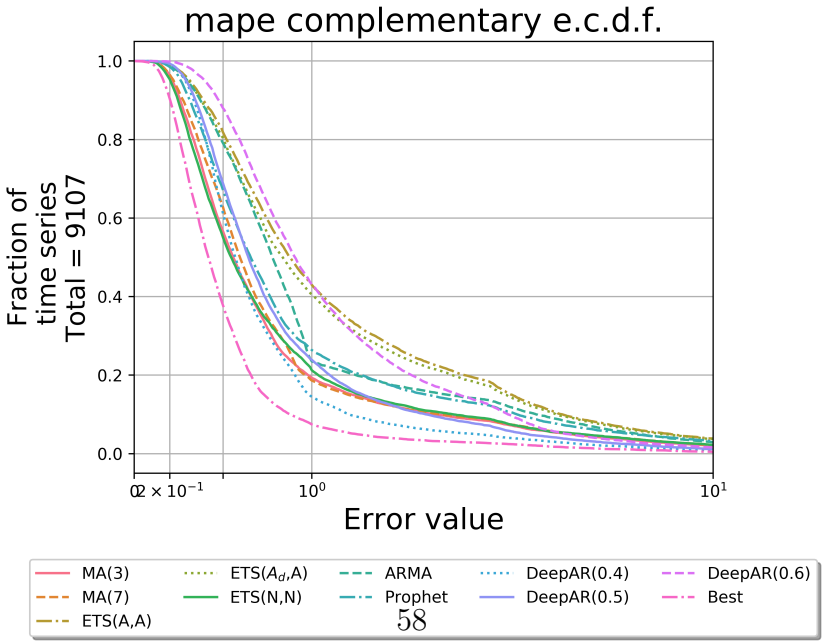
Tabela 6.7: *Ranking* de modelos para o intervalo mensal



(a) MAPE 1 dia

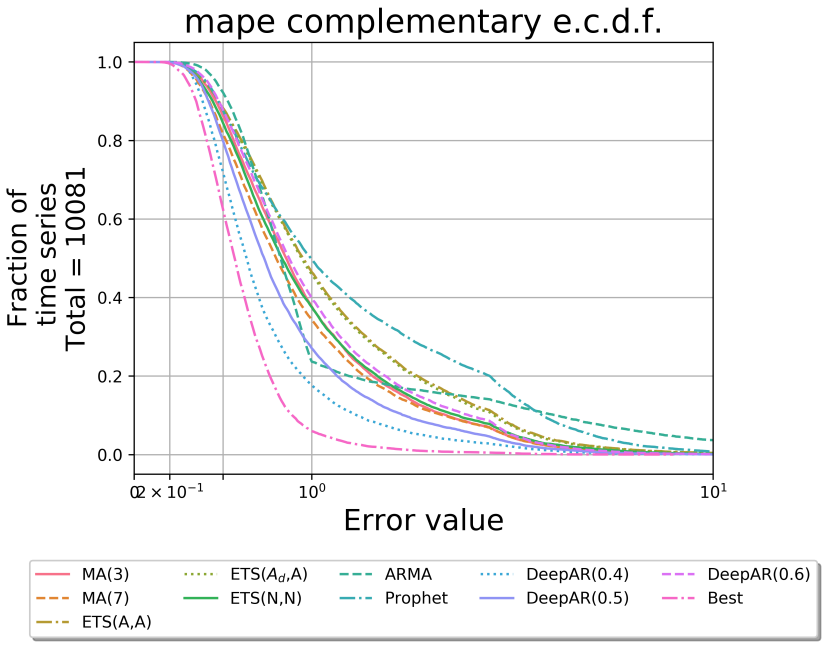


(b) MAPE 7 dias

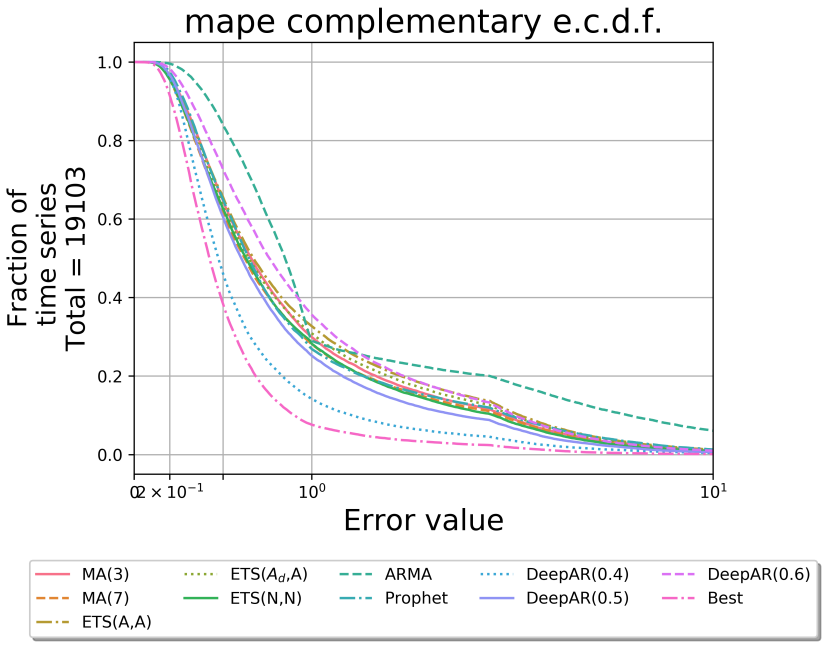


(c) MAPE 1 mês

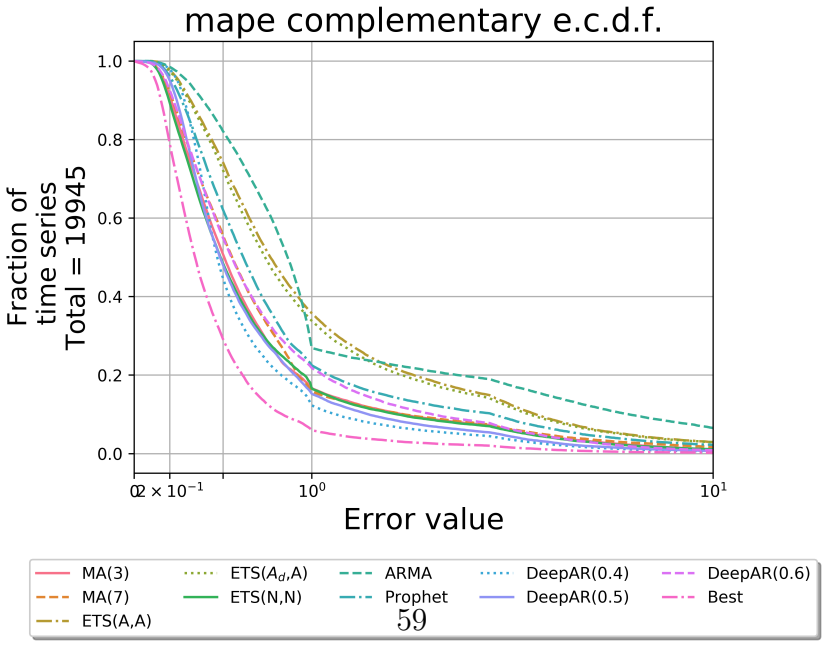
Figura 6.4: Distribuição do erro para o conjunto de produtos



(a) MAPE 1 dia

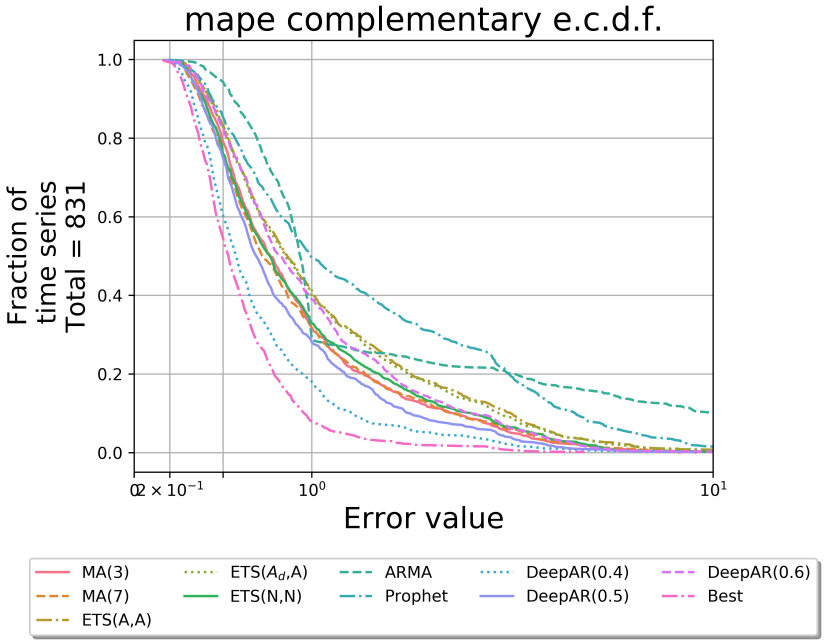


(b) MAPE 7 dias

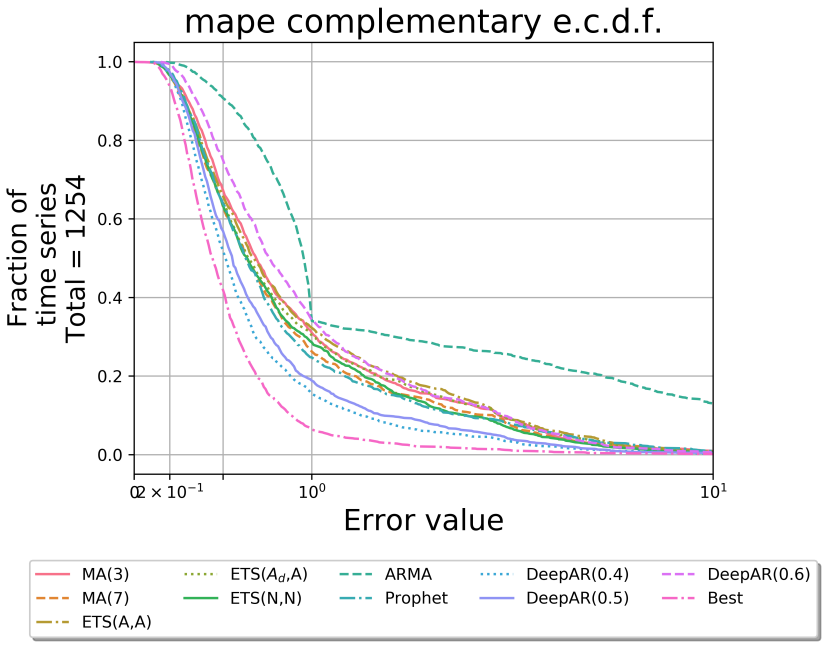


(c) MAPE 1 mês

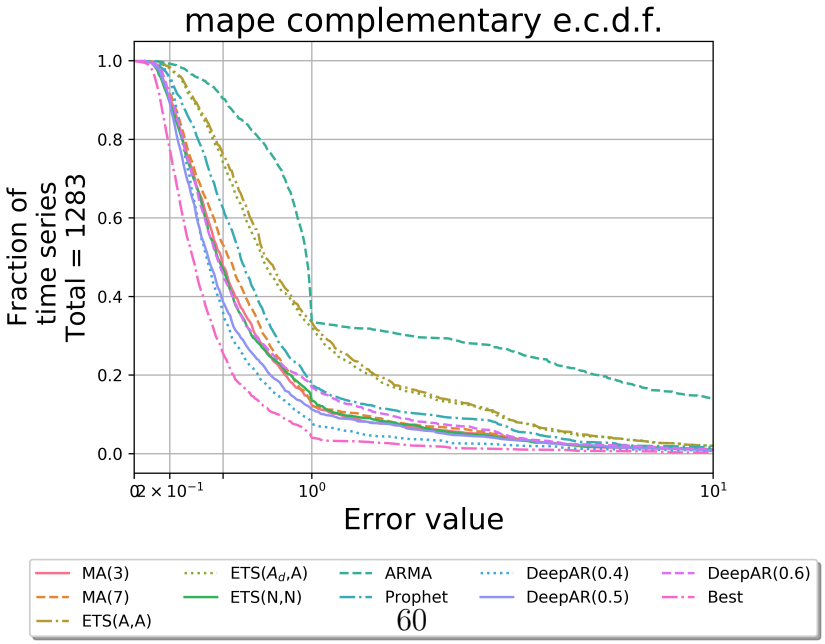
Figura 6.5: Distribuição do erro para o conjunto de categorias



(a) MAPE 1 dia

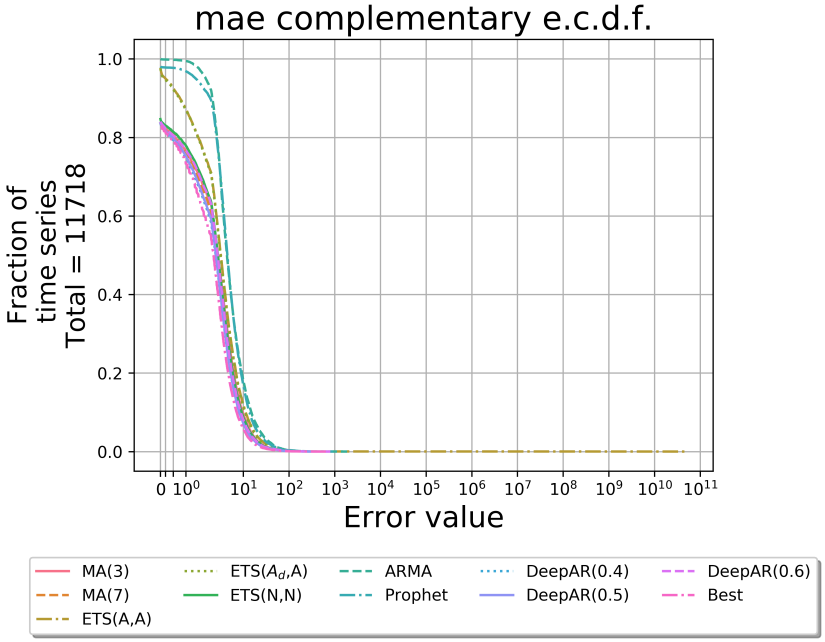


(b) MAPE 7 dias

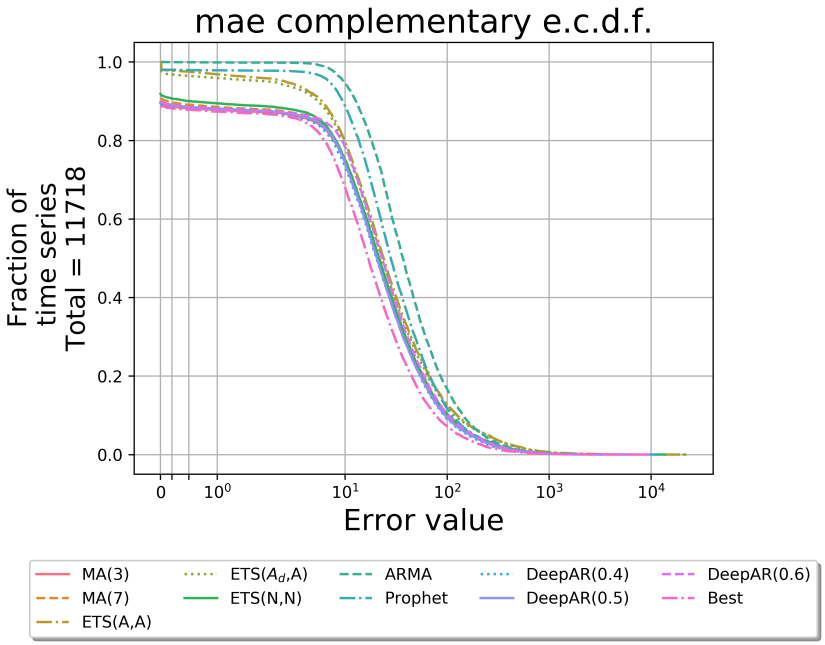


(c) MAPE 1 mês

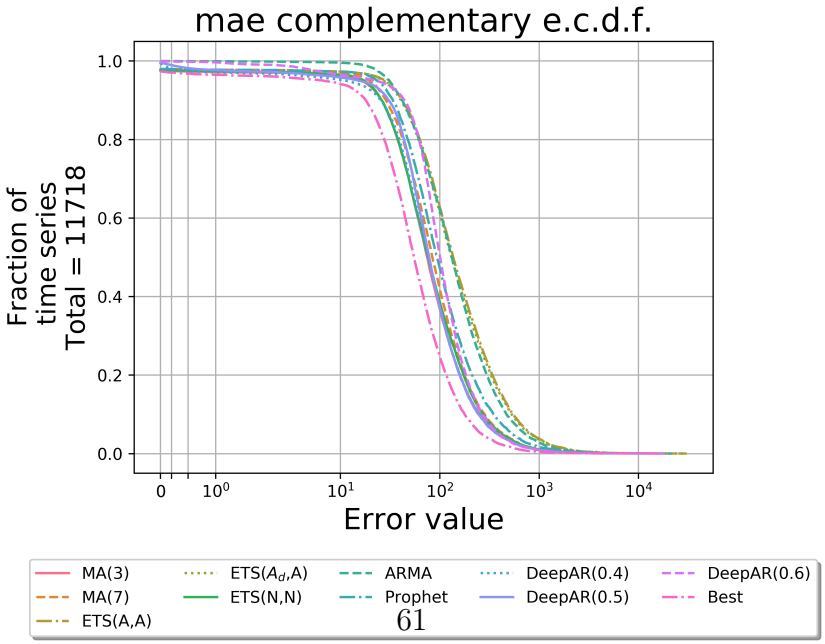
Figura 6.6: Distribuição do erro para o conjunto de contas



(a) MAE 1 dia

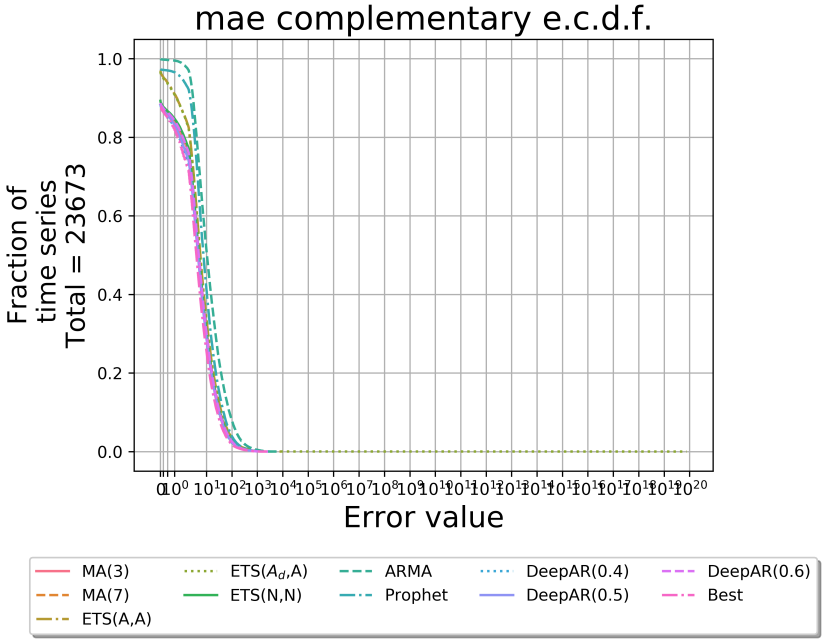


(b) MAE 7 dias

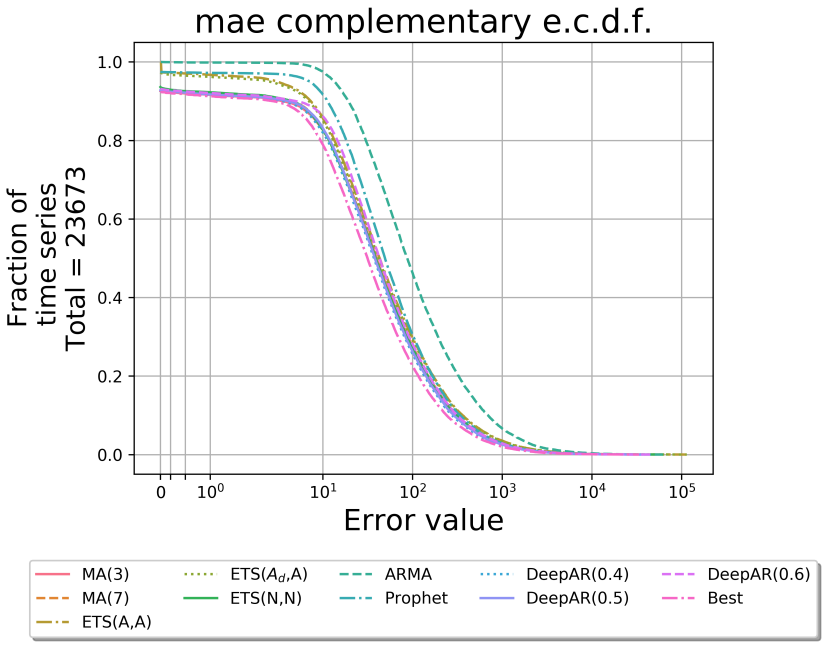


(c) MAE 1 mês

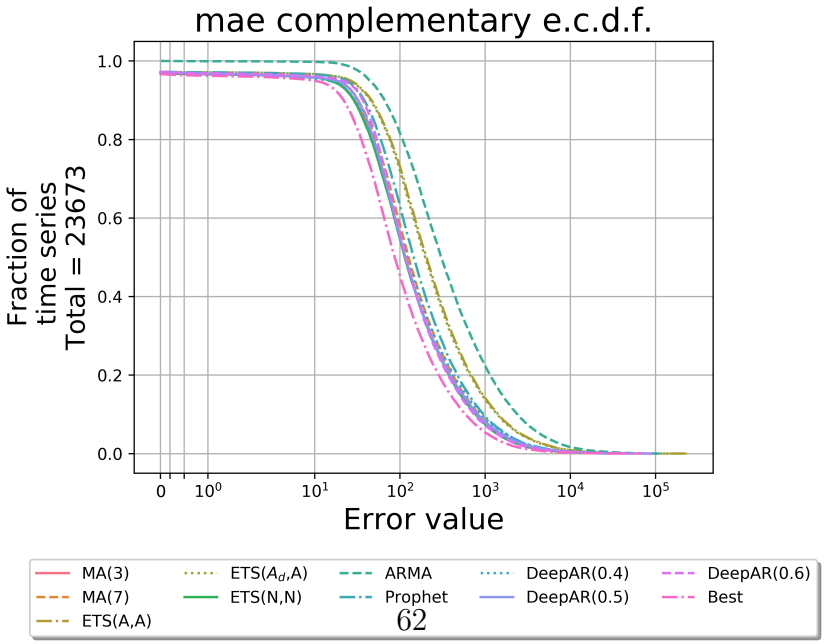
Figura 6.7: Distribuição do erro para o conjunto de produtos



(a) MAE 1 dia

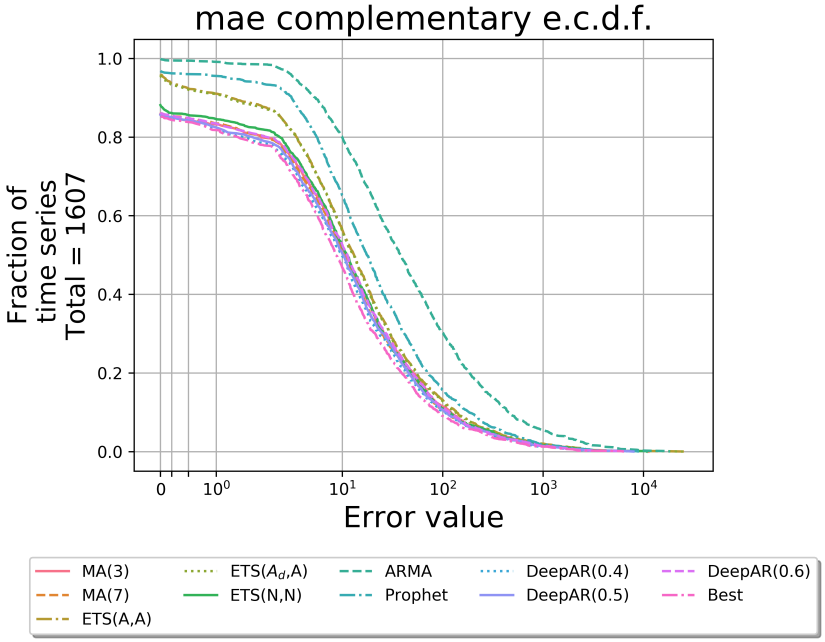


(b) MAE 7 dias

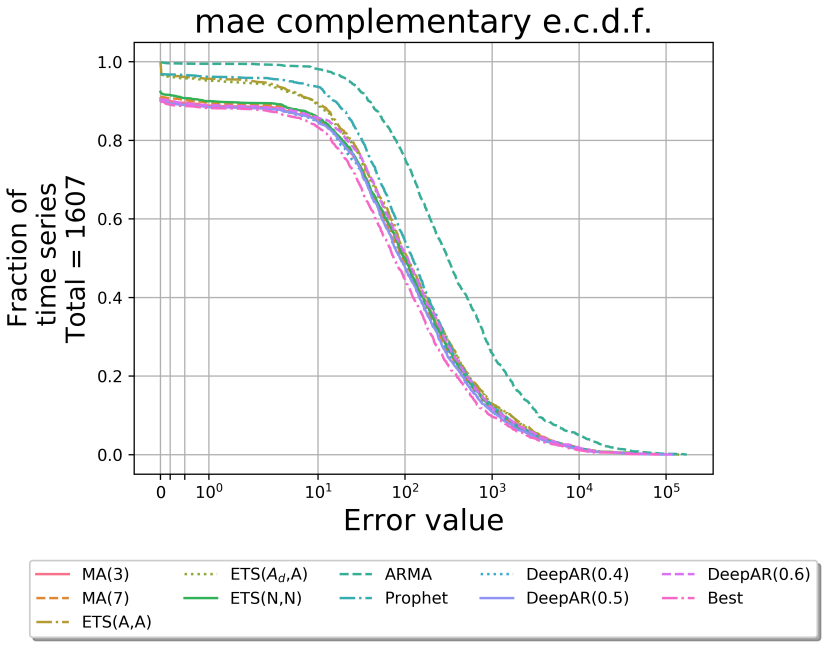


(c) MAE 1 mês

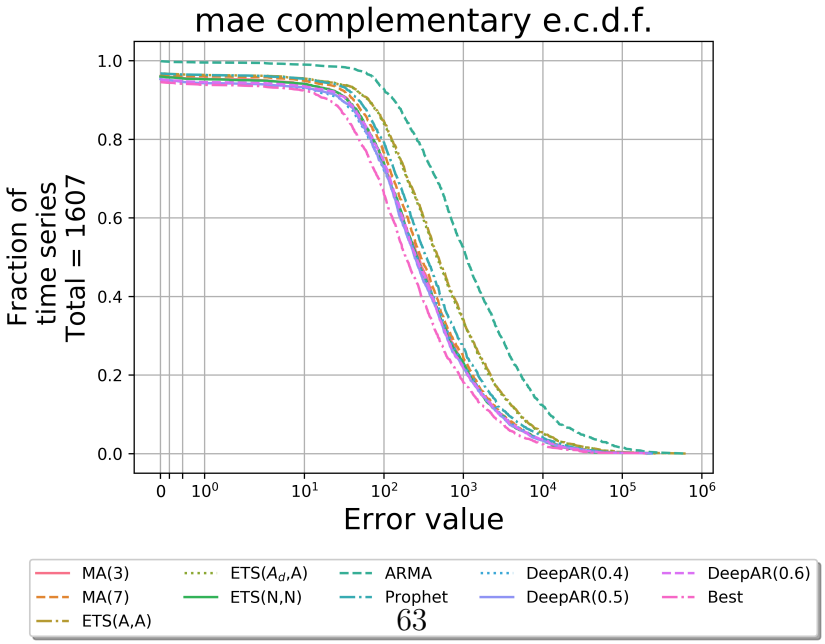
Figura 6.8: Distribuição do erro para o conjunto de categorias



(a) MAE 1 dia



(b) MAE 7 dias



(c) MAE 1 mês

Figura 6.9: Distribuição do erro para o conjunto de contas

As Figuras 6.4, 6.5 e 6.6 representam as funções de distribuições de probabilidade empíricas complementares dos erros de cada modelo para as métricas MAPE. Todos os gráficos estão em escala semi-log, o eixo X representa o erro MAPE ou MAE em escala logarítmica e o eixo Y a fração de séries temporais no intervalo $[0, 1]$. Todos os gráficos MAPE foram limitados para valores de erro no intervalo $[0, 1]$. Entre os aspectos gerais dos gráficos, podemos observar que o erro do método *Best* (linha rosa), por definição, está sempre abaixo de todos os outros modelos. As caudas das distribuições dos erros se tornam mais pesadas de produto para categoria e categoria para conta, com uma diferença clara entre produto e conta. Para alguns modelos, os erros caem mais rápido em segmentos com maior volume, por exemplo a estratégia *Best* no segmento de produtos na escala de tempo diária (Figura 6.4a) em relação ao mesmo modelo na escala de tempo mensal (Figura 6.4c). Podemos notar a grande variabilidade entre as magnitudes dos erros, por exemplo o segmento de produtos para a escala de tempo mensal (Figura 6.4c) no qual grande parte dos erros da métrica MAPE estão entre 10^1 e 10^3. Os gráficos de MAPE do modelo ARMA possui grande concentração em torno de 10^0 na maioria dos casos, artefato produzido pela realização de previsões de zeros. Uma outra peculiaridade associada a métrica MAPE é ser indefinida quando o valor real observado da série temporal é zero, o que faz os gráficos desta métrica possuírem diferentes quantidades de pontos em relação ao MAE. As Figuras 6.7, 6.8 e 6.9 representam as funções de distribuições de probabilidade empíricas complementares dos erros de cada modelo para as métricas MAE. A acumulação de erros fica clara entre as distribuições do MAE da Figura 6.7a contra o MAE da Figura 6.7c, no qual quase a totalidade das séries temporais estão abaixo de 10^1 vs. 10^3, respectivamente.

Podemos observar nos gráficos MAE que a cauda da distribuição possui erro (que depende da escala de tempo) muito elevado, uma estratégia para esses casos é limitar o provisionamento das previsões somente para os limiares de erros aceitáveis definidos pelos tomadores de decisão. Se utilizarmos a estratégia de melhor valor como referência, podemos observar que enquanto o segmento de produto para a escala de tempo de dia, semanal e mensal possuem médias de 5.72, 40.0 e 173.67 (Tabela 5.2) respectivamente. Temos que aproximadamente 20% dos erros MAE são maiores que 10 na escala de tempo diária (Figura 6.7a), 4% dos erros MAE maiores que 100 na escala de tempo semanal (Figura 6.7b) e 1% dos erros MAE maiores que 1000 na escala de tempo mensal (Figura 6.7c). Da mesma forma, para o segmento de categoria temos que as médias são 24.14, 168.77 e 731.49 (Tabela 5.3) e que 1% do erro MAE é maior que 100 na escala de tempo diária (Figura 6.8a), 1% do erro MAE maiores que 1000 na escala de tempo semanal (Figura 6.8b) e 1.5% maiores que 1000 na escala de tempo mensal (Figura 6.8c). Para o segmento de contas, para a escala de tempo de dia, semanal e mensal, temos que as médias são 137.36,

960.42 e 4165.66 (Tabela 5.4), respectivamente. Nesse segmento temos que 5% do MAE é maior que 100 na escala de tempo diária (Figura 6.9a), 6% maiores que 1000 (Figura 6.9b) e 1% maiores que 10000 (Figura 6.9c). Esses números confirmam o quão extremos são os erros em relação a média.

Na Figura 6.6c podemos observar que 80% das contas possuem um MAPE menor que 0.5 utilizando o critério *Best*. No segmento de categorias, este número diminui para 73% com MAPE menores que 0.5 (Figura 6.5c). No segmentos com menor volume - de produtos - o número diminui para 63% menores que 0.5 (Figura 6.4c). Da mesma forma para a métrica MAE, na Figura 6.9c podemos observar que 80% das contas tem erros menores que 1000 utilizando o critério *Best*. No segmento de categorias, este número diminui para 73% com MAE menores que 200 (Figura 6.8c). No segmentos com menor volume - de produtos - o número diminui para 63% menores que 9 (Figura 6.7c).

Embora o comportamento das curvas da distribuição MAPE e MAE de cada modelo sejam estáveis de modo geral, no qual um modelo mais acurado na média é superior também em distribuição, existem casos em que modelos trocam de posição. Um exemplo é a Figura 6.4a no qual os dois melhores modelos (excluindo a estratégia de melhor valor entre modelos) trocam de posição ao longo da distribuição: O modelo ARMA tem um melhor desempenho no intervalo $[0.9, 1.0]$ do que o melhor modelo na média DeepAR(0.4). Outro exemplo é o modelo DeepAR(0.4) na Figura 6.4c que se torna o melhor modelo em distribuição a partir de 0.6 de MAPE, antes deste ponto o modelo ETS(N,N) tem melhor desempenho. Nesta mesma Figura, o modelo DeepAR(0.6) embora tenha o pior resultado até MAPE 0.9, na cauda da distribuição ele torna-se melhor que ETS(A,A), ETS(A_d,A), ARMA e Prophet. Estes comportamentos se repetem nas Figuras 6.5 e 6.6. Isto indica que o desempenho dos modelos muitas vezes não pode ser aferido pela média, já que alguns modelos tem maior erro médio mas melhor desempenho para baixos valores de erro.

Capítulo 7

Conclusão

Produtos que são responsáveis por grande parte do faturamento dos clientes VTEX sofrem falta de estoque e não podem ser vendidos. Podemos observar na Figura 1.1 que praticamente toda a concentração das vendas ocorrem para três SKUs. Desta forma, é de fundamental importância o planejamento de inventário, com a previsão de demanda automatizada como um dos pilares do planejamento para evitar estas situações, aumentando o faturamento dos clientes.

As séries temporais utilizadas neste trabalho são provenientes de dados de fechamento de compras de clientes da VTEX no período de 01/01/2017 até 31/06/2019 (2 anos e 6 meses) que foram construídas dos dados brutos de fechamentos de pedido. Para o processamento dos dados foi utilizado o motor de processamento distribuído Apache Spark em um cluster com 408 vCPUs e 864 GiB de memória na plataforma de serviços de computação em nuvem AWS.

Diferentes tipos de agregações foram realizadas: produto, categoria e conta, do menor nível de segmentação para o maior nível, respectivamente. Para cada uma das segmentações, séries temporais em diferentes escalas de tempo foram criadas: por dia, por semana e por mês. Deste modo, obtemos 9 conjuntos de dados, conforme resumido na Tabela 5.1. Para cada série, novas séries temporais foram criadas a partir do truncamento das originais para a validação cruzada dos modelos: 14 novas séries para a escala de tempo diária; 8 novas séries para a escala semanal e 7 novas séries para a escala mensal. Somente séries temporais com volume de vendas total maiores que 2200 foram selecionadas para compor o conjunto de dados final, uma vez que a previsão de séries temporais com baixo volume de vendas tem pouco valor de negócio.

Nas Figuras 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.13, 5.14 e 5.15 observamos a diversidade de séries temporais analisadas através das e.c.c.d.f da soma, desvio padrão e média das séries temporais em diferentes segmentos e escala de tempo. Por exemplo, a Figura 5.9 mostra a e.c.c.d.f da média das séries temporais de produtos na escala de tempo mensal e que quanto maior o nível de agregação, maior a disparidade en-

tre as séries em todos os gráficos. Na Figura 5.15 de contas mensais, o nível mais suavizado devido a agregação, temos que o volume total de vendas para a maioria das séries gira em torno de 10^3 até 10^6, porém ainda existe uma fatia entre 0 e 10^3 e 10^6 até 10^7. Da mesma forma podemos observar uma grande variedade de desvios padrões, com concentração entre 10^1 até 10^4 e podemos até mesmo encontrar uma proporção de desvios padrões maiores que 10^4, como ilustrado na Figura 5.15.

A Figura 5.5 exibe quatro séries temporais de categorias de lojas diferentes individualmente. Podemos constatar que as séries temporais apresentam grande heteroscedasticidade, alternando em períodos de alta e baixa vendas pelas categorias, sendo a Figura 5.5d o maior exemplo deste comportamento. A Figura 5.6 mostra o nível de agregação por mês para produtos de duas lojas, embora os valores observados estejam suavizados por conta da escala de tempo mensal, os padrões das séries continuam não sendo muito claros.

Analisamos o desempenho de diversos modelos clássicos e modernos. Diversos métodos foram descartados das análises: modelos clássicos com variáveis exógenas, modelos hierárquicos, ARIMA, MQ-RNN, MQ-CNN e DeepFactor. Entre os modelos analisados, o modelo DeepAR mostrou-se superior em todos intervalos de tempo e segmentos. Enquanto a abordagem da utilização do melhor modelo para cada série obteve um resultado significativamente melhor do que a utilização do DeepAR para todas as séries temporais. Também notamos que o melhor modelo foi seguido de uma simples média móvel, o que significa que não necessariamente precisamos de grande poder computacional se a margem de erro aceitável for mais abrangente.

Neste trabalho foi realizado uma comparação de diversos modelos para previsão de séries temporais, no qual abordagens modernas que misturam abordagens locais e globais mostraram-se uma alternativa viável em relação aos modelos estatísticos tradicionais como ARIMA ou ETS. Vimos que a escolha natural do percentil 0.5 não obteve melhor desempenho: o melhor modelo, DeepAR(0.4), obteve erros MAPE de 0.74 em previsões para produtos diariamente até 1.21 para contas mensalmente. A estratégia de melhor valor obteve ganhos percentuais de 21.3% no intervalo diário até 80.6% em relação ao melhor modelo, ambos no segmentos de contas.

Observamos através das e.c.c.d.f. nas Figuras 6.4, 6.5 e 6.6 para a métrica MAPE e nas Figuras 6.7, 6.8, 6.9 que os erros possuem cauda pesada e que embora o comportamento das curvas da distribuição MAPE e MAE de cada modelo sejam estáveis de modo geral, no qual um modelo mais acurado na média é superior também em distribuição, existem casos em que modelos trocam de posição. Logo é necessário analisar toda a distribuição de erro para a escolha final do modelo a ser servidos aos clientes.

Em termos absolutos, a utilização de previsões de séries temporais para produtos de *e-commerce* demonstra um alto erro que tem como principal causa a grande

variedade de comportamento das séries, embora a utilização de maior poder computacional possa ser uma possível solução para viabilizar a entrega de previsões em larga escala de forma automática, seja através da incorporação de variáveis exógenas ou treinamento de hiper-parâmetros. Uma estratégia paliativa para esses casos é limitar o provisionamento das previsões somente para séries temporais que possuem limiares de erros aceitáveis (obtidos pela validação cruzada) definidos pelos tomadores de decisão.

Como trabalhos futuros, alguns caminhos valem a pena serem seguidos:

- Pré-processamento das séries temporais. O tratamento de todas as séries temporais foi realizada em conjunto e não apresentaram melhora significativa, porém é possível que diferentes pré-processamentos para diferentes séries temporais obtenham um melhor desempenho.

- Treinamento de hiper-parâmetros: devido a alta complexidade computacional o treinamento de hiper-parâmetros não foi realizado neste trabalho, isto pode melhorar consideravelmente os resultados.

- Incorporação de variáveis exógenas que podem contribuir para a acurácia do modelo, por exemplo: feriados, dados de navegação de usuários (como número de visualizações de um produto) e indicadores econômicos. Modelos globais como redes neurais podem incorporar novas variáveis de maneira mais eficiente.

- Uma quantificação de risco pode ser criada de forma a entregar previsões de acordo com a aversão ao risco de cada cliente e também de necessidades especificas. Vimos neste trabalho a importância da utilização de diferentes percentis, caso uma medida como o *coverage* (percentual de vezes que o modelo previu abaixo do real) seja relevante. Na medida em que quanto maior o percentil, maior o *coverage* de um conjunto de dados.

- Enquanto este trabalho foi desenvolvido, uma competição de previsão para vendas para o Walmart foi iniciado na plataforma Kaggle [15], ao fim da competição as técnicas e modelos ganhadores podem indicar novos caminhos a serem seguidos. O método vencedor até o momento, faltando 2 meses para a conclusão do desafio, é o LightGBM [45] que não foi explorado neste trabalho. LightGBM é um *framework* de *gradient boosting* que utiliza algoritmos de árvore de decisão para sua aprendizagem.

Referências Bibliográficas

[1] JAMES, G., WITTEN, D., HASTIE, T., et al. *An introduction to statistical learning*, v. 112. Springer, 2013.

[2] ABU-MOSTAFA, Y. S., MAGDON-ISMAIL, M., LIN, H.-T. *Learning from data*, v. 4. AMLBook New York, NY, USA:, 2012.

[3] ZHANG, A., LIPTON, Z. C., LI, M., et al. "Dive into Deep Learning", *Unpublished draft. Retrieved*, v. 3, pp. 319, 2019.

[4] HYNDMAN, R. J., ATHANASOPOULOS, G. *Forecasting: principles and practice*. OTexts, 2018.

[5] SIMCHI-LEVI, D., KAMINSKY, P., SIMCHI-LEVI, E., et al. *Designing and managing the supply chain: concepts, strategies and case studies*. Tata McGraw-Hill Education, 2008.

[6] HYNDMAN, R., KOEHLER, A. B., ORD, J. K., et al. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

[7] WANG, Y., SMOLA, A., MADDIX, D. C., et al. "Deep factors for forecasting", *arXiv preprint arXiv:1905.12417*, 2019.

[8] SALINAS, D., FLUNKERT, V., GASTHAUS, J., et al. "DeepAR: Probabilistic forecasting with autoregressive recurrent networks", *International Journal of Forecasting*, 2019.

[9] WANG, Y., SMOLA, A., MADDIX, D., et al. "Deep Factors for Forecasting". In: Chaudhuri, K., Salakhutdinov, R. (Eds.), *Proceedings of the 36th International Conference on Machine Learning*, v. 97, *Proceedings of Machine Learning Research*, pp. 6607–6617, Long Beach, California, USA, 09–15 Jun 2019. PMLR. Disponível em: <`http://proceedings.mlr.press/v97/wang19k.html`>.

[10] BÖSE, J.-H., FLUNKERT, V., GASTHAUS, J., et al. "Probabilistic demand forecasting at scale", *Proceedings of the VLDB Endowment*, v. 10, n. 12, pp. 1694–1705, 2017.

[11] VTEX. "VTEX - Accelerate Commerce Transformation". 2020. Disponível em: <`https://vtex.com/br-pt/`>. [Online; Accessed: 2020-04-21].

[12] WIKIPÉDIA. "VTEX — Wikipédia, a enciclopédia livre". 2018. Disponível em: <`https://pt.wikipedia.org/w/index.php?title=VTEX&oldid=53062853`>. [Online; Accessed: 2020-04-21].

[13] SILVER, E. A., PYKE, D. F., PETERSON, R., et al. *Inventory management and production planning and scheduling*, v. 3. Wiley New York, 1998.

[14] ZAHARIA, M., CHOWDHURY, M., FRANKLIN, M. J., et al. "Spark: Cluster computing with working sets." *HotCloud*, v. 10, n. 10-10, pp. 95, 2010.

[15] KAGGLE. "M5 Forecasting - Accuracy". 2020. Disponível em: <`https://www.kaggle.com/c/m5-forecasting-accuracy/`>. [Online; accessed 03-May-2020].

[16] ROSSUM, G. "Python reference manual", 1995.

[17] SEABOLD, S., PERKTOLD, J. "Statsmodels: Econometric and statistical modeling with python". In: *Proceedings of the 9th Python in Science Conference*, v. 57, p. 61. Scipy, 2010.

[18] ALEXANDROV, A., BENIDIS, K., BOHLKE-SCHNEIDER, M., et al. "GluonTS: Probabilistic Time Series Modeling in Python", *arXiv preprint arXiv:1906.05264*, 2019.

[19] BIANCHI, F. M., MAIORINO, E., KAMPFFMEYER, M. C., et al. "An overview and comparative analysis of recurrent neural networks for short term load forecasting", *arXiv preprint arXiv:1705.04378*, 2017.

[20] BROWN, R. G. *Statistical forecasting for inventory control*. McGraw/Hill, 1959.

[21] HOLT, C. C. "Forecasting seasonals and trends by exponentially weighted moving averages", *International journal of forecasting*, v. 20, n. 1, pp. 5–10, 2004.

[22] WINTERS, P. R. "Forecasting sales by exponentially weighted moving averages", *Management science*, v. 6, n. 3, pp. 324–342, 1960.

[23] GARDNER JR, E. S., MCKENZIE, E. "Forecasting trends in time series", *Management Science*, v. 31, n. 10, pp. 1237–1246, 1985.

[24] WILKS, S. S. "The large-sample distribution of the likelihood ratio for testing composite hypotheses", *The annals of mathematical statistics*, v. 9, n. 1, pp. 60–62, 1938.

[25] ROSENBLATT, F. "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, v. 65, n. 6, pp. 386, 1958.

[26] HORNIK, K. "Approximation capabilities of multilayer feedforward networks", *Neural networks*, v. 4, n. 2, pp. 251–257, 1991.

[27] RUMELHART, D. E., HINTON, G. E., WILLIAMS, R. J. "Learning representations by back-propagating errors", *nature*, v. 323, n. 6088, pp. 533–536, 1986.

[28] HOCHREITER, S. "The vanishing gradient problem during learning recurrent neural nets and problem solutions", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, v. 6, n. 02, pp. 107–116, 1998.

[29] HOCHREITER, S., SCHMIDHUBER, J. "Long short-term memory", *Neural computation*, v. 9, n. 8, pp. 1735–1780, 1997.

[30] FORECASTING, L.-R. "From Crystal Ball to Computer", *Scott Armstrong Robert J. Genetski*, 1978.

[31] ZAHARIA, M., XIN, R. S., WENDELL, P., et al. "Apache spark: a unified engine for big data processing", *Communications of the ACM*, v. 59, n. 11, pp. 56–65, 2016.

[32] WIKIPEDIA CONTRIBUTORS. "Extract, transform, load — Wikipedia, The Free Encyclopedia". 2020. Disponível em: <`https://en.wikipedia.org/w/index.php?title=Extract,_transform,_load&oldid=951295790`>. [Online; accessed 23-April-2020].

[33] VOHRA, D. "Apache parquet". In: *Practical Hadoop Ecosystem*, Springer, pp. 325–335, 2016.

[34] WHITE, T. *Hadoop: The definitive guide*. "O'Reilly Media, Inc.", 2012.

[35] "Amazon EC2 Instance Types". `https://aws.amazon.com/ec2/instance-types/`. Accessed: 2020-04-21.

[36] CLEVELAND, R. B., CLEVELAND, W. S., MCRAE, J. E., et al. "STL: A seasonal-trend decomposition", *Journal of official statistics*, v. 6, n. 1, pp. 3–73, 1990.

[37] VISWANATHAN, S., WIDIARTA, H., PIPLANI, R. "Forecasting aggregate time series with intermittent subaggregate components: top-down versus bottom-up forecasting", *IMA Journal of Management Mathematics*, v. 19, n. 3, pp. 275–287, 2008.

[38] BREUSCH, T. S., PAGAN, A. R. "A simple test for heteroscedasticity and random coefficient variation", *Econometrica: Journal of the Econometric Society*, pp. 1287–1294, 1979.

[39] SAKIA, R. M. "The Box-Cox transformation technique: a review", *Journal of the Royal Statistical Society: Series D (The Statistician)*, v. 41, n. 2, pp. 169–178, 1992.

[40] WU, Z., HUANG, N. E., LONG, S. R., et al. "On the trend, detrending, and variability of nonlinear and nonstationary time series", *Proceedings of the National Academy of Sciences*, v. 104, n. 38, pp. 14889–14894, 2007.

[41] TAYLOR, S. J., LETHAM, B. "Forecasting at scale", *The American Statistician*, v. 72, n. 1, pp. 37–45, 2018.

[42] HYNDMAN, R. J., KHANDAKAR, Y., OTHERS. *Automatic time series for forecasting: the forecast package for R*. N. 6/07. Monash University, Department of Econometrics and Business Statistics ..., 2007.

[43] WICKRAMASURIYA, S. L., ATHANASOPOULOS, G., HYNDMAN, R. J. "Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization", *Journal of the American Statistical Association*, v. 114, n. 526, pp. 804–819, 2019.

[44] WEN, R., TORKKOLA, K., NARAYANASWAMY, B., et al. "A multi-horizon quantile recurrent forecaster", *arXiv preprint arXiv:1711.11053*, 2017.

[45] KE, G., MENG, Q., FINLEY, T., et al. "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in neural information processing systems*, pp. 3146–3154, 2017.

preprocessing.py

```
1
2  def preprocess(raw):
3    df = raw.withColumn("exploded_items", F.explode(raw.Items)) \
4      .select(['creationDate', 'HostName', 'exploded_items.*']) \
5      .select(['creationDate', 'HostName', 'productId', '↘
           productCategories.id',
6                'quantity']) \
7      .withColumnRenamed('creationDate', 'timestamp') \
8      .withColumnRenamed('quantity', 'quantitySold') \
9      .withColumnRenamed('HostName', 'accountName') \
10     .withColumnRenamed('id', 'productCategoryArray') \
11     .withColumn('count', F.lit(1))
12
13   df = df.withColumn("timestamp", df['timestamp'].cast(T.↘
        TimestampType())) \
14     .withColumn("count", df['count'].cast(T.DoubleType())) \
15     .withColumn("quantitySold", df['quantitySold'].cast(T.↘
          DoubleType()))
16
17   return df
```

ETL.py

```
1
2  sum_by_fields = ['productId', 'productCategoryId']
3  freqs = ['1 day', '1 month', '7 days']
4  prediction_lengths = [1, 1, 1]
5  number_of_partitions = 800
6  sum_filter = 2200.0
7  today_str = datetime.datetime.now().strftime('%Y_%m_%d_%H_%M_%S')
8
9  spark = SparkSession \
10   .builder \
11   .appName("Demand Planning ETL") \
12   .getOrCreate() \
13   .conf.set('spark.sql.session.timeZone', 'UTC')
14
15 begin_year = 2017
16 end_year = 2019
17 begin_month = 1
18 end_month = 6
19
20 start_date = datetime.datetime(begin_year, begin_month, 1, 0, 0)
21 start_date = start_date + relativedelta(weekday=MO(-1))
22 end_test = datetime.datetime(end_year, end_month, 30, 0, 0)
23 end_test = end_test + relativedelta(weekday=MO(1))
```

```
24
25 raw = spark.read.parquet(
26   's3://vtex-ml/demand-planning/data/stage/2017_01_to_2019_06/*') \
27   .repartition(number_of_partitions, "Items.productId")
28
29
30 def process(raw, _key, idx, freq, use_dynamic_feat=False):
31   # Don't explode categories
32   if ((set(['accountName']) == set(_key)) or 'productId' in _key):
33     df = raw
34   else:
35     df = raw.withColumn('exploded_categories',
36                         F.explode('productCategoryArray')) \
37       .withColumnRenamed('exploded_categories', 'productCategoryId'↘
            )
38
39   df = df.withColumn("timestamp", F.date_trunc(truncate_by_freq,
40                                                df['timestamp'].cast↘
                                                    (
41                                                  T.TimestampType())↘
                                                      ))
42
43   if 'productId' in _key:
44     df = df.withColumn('item_id', F.concat(F.col('accountName'), F.↘
          lit('_'),
45                                            F.col('productId')))
46   elif 'productCategoryId' in _key:
47     df = df.withColumn('item_id', F.concat(F.col('accountName'), F.↘
          lit('_'),
48                                            F.col('productCategoryId↘
                                                ')))
49   else:
50     df = df.withColumn('item_id', F.col('accountName'))
51
52   _key = _key + ['item_id']
53
54   aggregation = {
55       'quantitySold': 'sum',
56       'count'       : 'sum'
57   }
58
59   if 'month' not in freq:
60     _aux = _key + [F.window("timestamp", freq, startTime='{} days'.↘
          format(
61           offset_days))]
62   else:
63     _aux = _key + ['timestamp']
```

```
64
65    df_sum_by_field = df.groupBy(_key).agg({'count': 'sum'})
66    df_sum_by_field = df_sum_by_field.filter(
67      df_sum_by_field['sum(count)'] > sum_filter)
68    top_df = df.join(df_sum_by_field, _key, "right")
69    aggregated_df = top_df.groupBy(_aux).agg(aggregation)
70
71    if 'month' not in freq:
72      aggregated_df = aggregated_df.select("window.*", "*").drop(
73        'window').withColumnRenamed("start",
74                                    "timestamp").drop("end")
75
76    epoch = (aggregated_df["timestamp"].cast("bigint") / multiplier).↘
          cast(
77      "bigint") * multiplier
78    with_epoch = aggregated_df.withColumn("epoch", epoch)
79
80    if 'month' not in freq:
81      ref_test = spark.range(
82          min_epoch, max_epoch_test + 1, multiplier
83      ).toDF("epoch") \
84        .withColumn("quantitySoldRef", F.lit(0.0))
85
86      ref_test = ref_test.withColumn("timestamp",
87                                     F.date_trunc(truncate_by_freq,
88                                                  ref_test['epoch'].↘
                                                       cast(
89                                                    T.TimestampType()↘
                                                       )))
90      ref_test = ref_test.withColumn("epoch",
91                                     (ref_test["timestamp"].cast(
92                                       "bigint") / multiplier).cast(
93                                       "bigint") * multiplier) \
94        .drop('timestamp')
95    else:
96      dates = [(start_date, end_test)]
97      aux = spark.createDataFrame(dates, ["minDate", "maxDate"])
98      ref_test = aux.withColumn("monthsDiff",
99                                F.months_between("maxDate", "minDate"↘
                                     )) \
100       .withColumn("repeat", F.expr("split(repeat(',', monthsDiff), ↘
              ',')")) \
101       .select("*", F.posexplode("repeat").alias("date", "val")) \
102       .withColumn("date", F.expr("add_months(minDate, date)")) \
103       .withColumn('date', F.date_trunc(truncate_by_freq, F.col('↘
              date'))) \
104       .withColumn('epoch', F.col('date').cast("bigint")) \
```

```
105         .select('epoch') \
106         .withColumn('quantitySoldRef', F.lit(0.0))
107
108     products_df = aggregated_df.select(_key).distinct()
109     cross_joined_test = ref_test.crossJoin(products_df)
110     _aux = _key + ["epoch"]
111     ans_test = cross_joined_test.join(with_epoch, _aux, "left")
112     test_data = ans_test.withColumn("quantitySoldAfterJoins",
113                                     F.when(ans_test['sum(quantitySold↘
                                           )'].isNull(),
114                                            ans_test['quantitySoldRef'↘
                                               ]).otherwise(
115                                           ans_test['sum(quantitySold)'↘
                                               ])) \
116       .withColumn("countAfterJoins",
117                   F.when(ans_test['sum(quantitySold)'].isNull(),
118                          ans_test['quantitySoldRef']).otherwise(
119                       ans_test['sum(count)']))
120
121     test_data = test_data.withColumn("timestamp",
122                                      F.col('epoch').cast(T.↘
                                           TimestampType()))
123     test_data = test_data.withColumn("timestamp",
124                                      F.from_unixtime(
125                                        F.unix_timestamp(test_data['↘
                                             timestamp']),
126                                        "yyyy-MM-dd HH:mm:ss"))
127     aggregated_lists_test_with_start_date = test_data.↘
          withColumnRenamed(
128       "countAfterJoins", "demand")
129     aggregated_lists_test_with_start_date = \
130         aggregated_lists_test_with_start_date.select(
131       'item_id', 'timestamp',
132       'demand')
133
134     return aggregated_lists_test_with_start_date
135
136
137 _key_account = ['accountName']
138
139 # aggregation by account and products (or categories, brands, ...)
140 for idx, freq in enumerate(freqs):
141
142     for sum_by_field in sum_by_fields:
143       _key = _key_account + [sum_by_field]
144       aggregated_lists_test_with_start_date = process(raw, _key, idx,↘
              freq,
```

```
145                                                     use_dynamic_feat↘
                                                            )
146
147       save_to_s3(aggregated_lists_test_with_start_date, _key, 'test',
148                  sum_by_field)
149
150   #     aggregation by account
151   aggregated_lists_test_with_start_date = process(raw, _key_account↘
          , idx, freq,
152                                                   use_dynamic_feat)
153
154   _key = _key_account
155   save_to_s3(aggregated_lists_test_with_start_date, _key, 'test',
156              _key_account[0], use_dynamic_feat)
```