UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Leandro Vargas Barbosa

MODELO DE PREVISÃO DE CONSUMO DE ENERGIA ELÉTRICA EM CENÁRIO PANDÊMICO COM USO DE INTELIGÊNCIA ARTIFICIAL

Porto Alegre

10/11/2021

Leandro Vargas Barbosa

MODELO DE PREVISÃO DE CONSUMO DE ENERGIA ELÉTRICA EM CENÁRIO PANDÊMICO COM USO DE INTELIGÊNCIA ARTIFICIAL

Orientador(a): Profª. Drª. Gladis Bordin

Porto Alegre
10/11/2021

Leandro Vargas Barbosa

MODELO DE PREVISÃO DE CONSUMO DE ENERGIA ELÉTRICA EM CENÁRIO PANDÊMICO COM USO DE INTELIGÊNCIA ARTIFICIAL

Projeto de Diplomação apresentado ao Curso de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como requisito parcial para Graduação em Engenharia Elétrica.

Orientador(a): Profª. Drª. Gladis Bordin

Porto Alegre

2021

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO

CIP - Catalogação na Publicação

Barbosa, Leandro Vargas
MODELO DE PREVISÃO DE CONSUMO DE ENERGIA ELÉTRICA EM CENÁRIO PANDÊMICO COM USO DE INTELIGÊNCIA ARTIFICIAL / Leandro Vargas Barbosa. -- 2021.
213 f.
Orientadora: Gládis Bordin.

Trabalho de conclusão de curso (Graduação) -- Universidade Federal do Rio Grande do Sul, Escola de Engenharia, Curso de Engenharia Elétrica, Porto Alegre, BR-RS, 2021.

1. Previsão de Consumo. 2. Energia Elétrica. 3. Inteligência Artificial. 4. COVID19. 5. Redes Neurais. I. Bordin, Gládis, orient. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da UFRGS com os dados fornecidos pelo(a) autor(a).

Leandro Vargas Barbosa

MODELO DE PREVISÃO DE CONSUMO DE ENERGIA ELÉTRICA EM CENÁRIO PANDÊMICO COM USO DE INTELIGÊNCIA ARTIFICIAL

Projeto de Diplomação apresentado ao Curso de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como requisito parcial para Graduação em Engenharia Elétrica.

Aprovado em: 26 de outubro de 2021.

BANCA EXAMINADORA

_____

Prof. Dr. Roberto Petry Homrich

Universidade Federal do Rio Grande do Sul

_____

Prof. Dr. Fabio Souto De Azevedo

Universidade Federal do Rio Grande do Sul

_____

Profª. Drª. Gladis Bordin

Universidade Federal do Rio Grande do Sul

AGRADECIMENTOS

Dedico esse trabalho a minha amada Mãe Ivone Vargas (*in memoriam*) que sempre me incentivou e acreditou em mim, e certamente estaria muito feliz de ver este sonho concretizado. A minha irmã Franciele Vargas Barbosa (*in memoriam*) que tão breve nos deixou, seria tão bom partilhar contigo este momento. Saudade Eterna.

Dedico também a meu Pai, José da Costa Barbosa, que igualmente me apoio e sempre me aconselhou a trilhar os caminhos com respeito honra e honestidade. A minha querida Irmã Fernanda Vargas Barbosa, saudades dos tempos que devíamos ter passado juntos e conversas que não pudemos ter devido ao pouco tempo que tive presente.

A minha sogra Laura Solange Gonçalves Dias, pela grande ajuda principalmente dando carinho e atenção ao meu filho seu neto suprindo minha ausência, durante a realização deste projeto.

Em especial a minha amada companheira Lauriane Dias Freitas, que dividiu comigo boa parte dessa jornada de engenharia, e a meu amado filho Guilherme Freitas Barbosa, não chegaria até aqui sem vocês, sem seu apoio, sem seu carinho, sem sua compreensão e sem seu amor.

Aos mestres por todo aprendizado e ensinamentos transmitidos, em especial a Profª. Drª. Gladis Bordin que me ajudou na construção deste trabalho e sempre foi um exemplo e inspiração como grande ser humano que és.

RESUMO

O trabalho propõe um novo modelo estocástico de projeção de consumo de energia elétrica, utilizando técnicas de otimização através de algoritmo genético aplicados a uma rede neural recorrente do tipo *Long Short Term Memory*. Ressalta o efeito das mortes por COVID-19 no consumo de energia, e contempla o efeito aleatório e de atenuação de um transitório social, o que costumeiramente é simplificado em modelos de projeção de consumo. O modelo construído é apresentado detalhadamente usando, inicialmente, dados do Rio Grande do Sul e, posteriormente, amplia sua abrangência em um estudo de caso para o Brasil em cenário pré-pandemia em 2019 e em cenário pandêmico em 2020. Os testes de variáveis econômicas climáticas e sociais são realizados e mostram que as técnicas utilizadas são adequadas à representação do problema sob análise.

Palavras-chave: Previsão de Consumo, Energia Elétrica, Inteligência Artificial, COVID19.

ABSTRACT

This paper proposes a new stochastic model for projecting electricity consumption, using optimization techniques through genetic algorithms applied to a recurrent neural network of the Long Short Term Memory type. It highlights the effect of COVID-19 deaths on energy consumption, and considers the random and attenuation effect of a social transient, which is usually simplified in consumption projection models. The model built is presented in detail initially using data from Rio Grande do Sul (southern state of Brazil) and later extends its scope in a case study to Brazil in a pre-pandemic scenario in 2019 and in a pandemic scenario in 2020. Tests of economic-climate and social variables are performed and show that the techniques used are adequate to represent the problem under analysis.

Keywords: Consumption Forecasting, Electricity, Artificial Intelligence, COVID19

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

AEMO	-	*"Australian Energy Market Operator"*, Operador do Mercado Australiano de Energia,
ANFIS	-	*"Adaptive Neuro-Fuzzy Inference Sistem",* Rede Neuro Fuzzy
API	-	*"Application Programming Interface"*, Interface de Programação de Aplicações
B3	-	Brasil Bolsa Balcão
BCB	-	Banco Central do Brasil
CNN	-	*"Convolutional Neural Network",* Rede Neural Convolucional
COVID	-	*"Corona Virus Disease",* Doença do Coronavírus
CPU	-	*"Central Process Unit",* Unidade Central de Processamento
EPE	-	Empresa de Pesquisa Energética
EPICOVID	-	Epidemiologia da Covid-19
FRBF	-	*"Fuzzy Radial Basis Linear",* Fuzzy Radio Base Linear
FUZZY	-	Método de Inferência FUZZY, Lógica Fuzzy
GPU	-	*"Graphics Processing Unit"*, Unidade de Processamento Gráfico
IBC-BR	-	Índice de Atividade Econômica do Banco Central – Brasil
IBOV	-	Índice da Bolsa de Valores
IMFs	-	Funções de modo intrínsecas
IPCA	-	Índice Nacional de Preços ao Consumidor Amplo
LSTM	-	*"Long Short-Term Memory Network",* Longa memória de curto prazo
LTLF	-	*"Long-term Load Forecasting",* Previsão de carga a longo prazo
MAPE	-	*"Mean Absolute Percentage Error",* Média do Erro Absoluto
MTLF	-	*"Medium-term Load Forecasting",* Previsão de carga a médio prazo
MWh	-	mega watt-hora
NN	-	*"Neural Network",* Rede Neural
PMU	-	*"Phasor Measurement Unit",* Unidade de Medição Fasorial / Síncrono Fasor
RMSE	-	*"Root Mean Squared Error",* Raiz Quadrada do Erro Quadrado Médio
RNN	-	*"Recurrent neural network",* Redes Neurais Recorrentes
SCADA	-	*"Supervisory Control and Data Acquisition",* Sistemas de Supervisão e Aquisição de
SRAG	-	Síndrome Respiratória Aguda Grave
STLF	-	*"Short-term Load Forecasting",* Previsão de carga a curto prazo
SUS	-	Sistema Único de Saúde

TPU	-	*"Tensor Processing Unit",* Unidade de Processamento de Tensor
UFRGS	-	Universidade Federal do Rio Grande do Sul
WLS	-	*"Weighted least square state estimation",* Estimativa por método dos mínimos quadrados.

SUMÁRIO

1 INTRODUÇÃO

Um problema relevante e complexo relacionado ao planejamento da gestão empresarial de uma concessionária de energia elétrica diz respeito à projeção do consumo de energia elétrica. Sabe-se que as projeções, em geral, não constituem um fim, mas um instrumento que fornece informações para uma consequente tomada de decisão com objetivos específicos. Assim, a projeção de consumo visa atender às necessidades dos consumidores com um nível de qualidade satisfatório.

A base de todo o processo de planejamento da expansão e operação de um sistema elétrico está ancorada em suas projeções de demanda por consumo de energia elétrica, tanto no que se refere aos aspectos técnicos quanto aos aspectos financeiros, como: o planejamento da geração, a transmissão e distribuição, o planejamento da operação e o planejamento financeiro.

A demanda de energia elétrica se reflete diretamente na atividade econômica, e, consequentemente, no desenvolvimento de uma região ou de um país. Assim sendo, a previsão do consumo de energia é uma informação basilar na decisão de investimento de qualquer ramo empresarial.

A projeção usual de consumo de energia elétrica baseia-se principalmente em técnicas econométricas, através de correlações entre a série histórica do consumo de energia elétrica e indicadores sócio-econômicos. No entanto, com o evento mundial da pandemia do COVID-19 – *Corona Virus Disease* (Doença do Coronavírus) – e seus efeitos sobre o consumo, uma vez que dados medidos de demanda atendida após o início da pandemia mostram a forte correlação entre o evento e a evolução do consumo, se faz necessário repensar os modelos de projeção de consumo em uso e adequá-los a esse novo contexto, buscando tornar as projeções mais realísticas para tomada de decisão de investimento

Neste contexto, o presente trabalho propõe um modelo estocástico de projeção de consumo de energia elétrica, com uso de ferramentas e técnicas de *machine learning* de código aberto implementadas em linguagem de programação *phyton* [1] para modelar o efeito do evento aleatório COVID-19 sob as projeções de consumo de médio e longo prazo. O modelo proposto faz uso de indicadores econômicos tradicionais, entre outros, como IPCA (Índice Nacional de Preços ao Consumidor Amplo), indicadores de mercado como IBOV (Índice da Bolsa de Valores), IBC-BR (Índice de Atividade Econômica do Banco Central – Brasil) e indicadores sociais e de saúde como registro de óbitos (TABNET SUS), contrastando e

correlacionando os indicadores citados com os dados históricos de consumo disponibilizados pelo anuário Empresa de Pesquisa Energética - EPE. Os cenários de consumo são projetados através de uma rede neural recorrente do tipo LSTM – *Long Short Term Memory*.

O presente trabalho se diferencia de estudos anteriores tanto na técnica quanto no enfoque de modelagem. Neste sentido, utiliza uma abordagem de vanguarda no emprego de inteligência artificial e explora variáveis não tradicionais, ou seja, o efeito da pandemia de COVID-19 no consumo de energia elétrica no Brasil. Ainda, o modelo desenvolvido contempla o que costumeiramente é simplificado em modelos de projeção de consumo de energia elétrica: o efeito aleatório e de atenuação de um transitório social.

1.1 Motivação do Trabalho

Um problema relevante e complexo relacionado ao planejamento da gestão empresarial de uma concessionária de energia elétrica diz respeito à projeção do consumo de energia elétrica. Sabe-se que as projeções, em geral, não constituem um fim, mas um instrumento que fornece informações para uma consequente tomada de decisão, com objetivos específicos. Assim, a projeção de consumo de energia elétrica visa atender às necessidades dos consumidores, com um nível de qualidade satisfatório.

A base de todo o processo de planejamento da expansão e operação de um sistema elétrico está ancorada nas suas projeções de demanda por consumo de energia elétrica, tanto no que se refere aos aspectos técnicos quanto aos aspectos financeiros, como: os planejamentos da geração, da transmissão e da distribuição; o planejamento da operação e o planejamento financeiro.

Ainda, a demanda de energia elétrica se reflete diretamente na atividade econômica e industrial, e a falta de energia é um dos fatores fundamentais na limitação do desenvolvimento e crescimento de uma região ou de um país. Assim sendo, a previsão desse consumo de energia é uma informação basilar na decisão de investimento de qualquer ramo empresarial.

1.2 O Problema

No processo de planejamento tradicional, a projeção de consumo de energia elétrica de curto prazo baseia-se principalmente em técnicas econométricas, através de correlações entre a série histórica da demanda de energia elétrica e indicadores sócio-econômicos. Para o horizonte de longo prazo, a projeção de demanda é realizada com uso da técnica de cenários sob incerteza.

No entanto, com o evento mundial da pandemia da COVID e seus efeitos sobre a este consumo, uma vez que dados medidos de demanda atendida após o início da pandemia mostram

a forte correlação entre o evento e a evolução da demanda, se faz necessário repensar os modelos de projeção de consumo de energia em uso e adequá-los e esse novo contexto, buscando tornar as projeções mais realísticas para futuras tomadas de decisão de investimento.

Inicialmente, utiliza uma abordagem simplificada, avaliando o histórico de consumo de energia e iterações com variáveis climáticas. Em seguida é agregada complexidade utilizando indicadores econômicos como: IPCA (Índice Nacional de Preços ao Consumidor Amplo), IBOV (Índice da Bolsa de Valores), IBC-Br (Índice de Atividade Econômica do Banco Central). E por fim é avaliado o uso do número de óbitos registrados no SUS (Sistema Único de Saúde).

1.3 Considerações sobre consumo e projeção de energia Elétrica

De acordo com a EPE (Empresa de Pesquisa Energética) [2], o consumo de energia elétrica é o consumo de eletricidade ao final de um período (dia, mês, ano, etc.) normalmente (para grandes sistemas) medidos em mega watt-hora. Nesse estudo usa-se mega watt-hora MWh.

Nesse trabalho, o interesse reside em prever o comportamento do consumo ao longo dos dias, meses, anos, através da correlação como a atividade econômica correspondente. Desconsiderando, numa abordagem inicial, a possível perda de demanda por parte da concessionária devido à autoprodução, às perdas técnicas e comerciais.

Entende-se por projeção de consumo de energia, uma estimativa estatística de como se esperam os níveis de consumo em um momento futuro.

Uma vez que o consumo evolui de forma diferente nos distintos tipos de consumidores, é usada a segmentação em classes de consumo aplicada no anuário da EPE [2], ou seja, Residencial, Industrial, Comercial, Rural, Poder Público, Iluminação Pública, Serviço Público, Consumo Próprio.

Após analisar alguns trabalhos, constata-se que em sua maioria bem como no próprio relatório oficial produzido pela EPE, baseiam se em projeções enviadas pelas distribuidoras de energia, estas por sua vez, se baseiam em estudos locais de projeção, demandas previstas e contratadas pelos consumidores e expectativas públicas crescimentos ou (e.g.) implantação de zona industrial. Os quais podem conter metas de políticas econômicas governamentais, que podem ou não serem atingidas. Este estudo propõem uma visão alternativa, valendo-se do histórico de demanda por região (da mesma forma que o estudo oficial), mas correlacionando-os diretamente aos históricos de mercado obtendo as próprias de projeções. Em especial,

pretende-se observar, com enfoque especial, o evento aleatório COVID19 que não está contemplado em nenhum destes estudos, conforme a revisão bibliográfica a seguir.

1.4 Revisão bibliográfica

No que segue, descreve-se os trabalhos mais relevante entre os analisados e algumas discussões relevantes sobre o tema.

- *Next Day Load Demand Forecasting of Future in Electrical Power Generation on Distribution Networks using Adaptive Neuro-Fuzzy Inference* [3].

Este artigo relata o desenvolvimento de método baseado em Neuro FUZZY (ANFIS-*Adaptive Neuro-Fuzzy Inference Sistem*) com o desenvolvimento de uma rede neural adaptativa para previsão de demanda diária para o dia seguinte. Para o protótipo utilizou-se dados de 24 dias do carregamento diário/hora de uma subestação de Bangkok. As entradas são Carga do dia atual, Carga do dia similar anterior e Previsão de Temperatura. A saída fornece a Previsão de carga máxima para cada hora e o erro médio obtido de 1,5%.

Apesar do erro médio baixo, houve 4 ocorrências com mais de 9,7% de erro calculando o erro quadrático médio encontra-se 5,33 e apenas 46% das amostras ficaram abaixo desse valor. A apresentação de apenas a média dos erros foi uma análise tendenciosa.

Considerando que a rede neural adaptativa conseguiu obter o melhor ajuste da rede, talvez o uso de mais entradas possa refinar os resultados.

- *Load Forecasting Through Estimated Parametrized Based Fuzzy Inference System in Smart Grids* [4].

Este artigo relata o desenvolvimento de método baseado em Neuro FUZZY(ANFIS + WLS* + NN*) para predição de carga de um sistema em grid de distribuição de energia de 14 e 30 barras, com Entradas a Tensão, a Frequência e a Corrente; e Saída a Previsão de Carga (obtendo com um erro quadrado médio de 2,9%).

Este trabalho está melhor fundamentado e, possivelmente utilizando o mesmo método, o resultado pudesse ser refinado com o uso de mais entradas, o que subverteria um pouco a ideia de simplicidade e performance defendida pelo autor no início do trabalho.

*obs.: WLS – Ajuste de pesos por método dos mínimos quadrados, do inglês: “Weighted least square state estimation”;

**obs.: NN – Redes Neurais, do ingês:“Neural Network”

- *A Reliability-Oriented Fuzzy Stochastic Framework in Automated Distribution Grids to Allocate* $\mu - PMUs$ [5]

Relata o desenvolvimento de um *FRAMEWORK* baseado em FUZZY (Monte Carlo) + Micro-Sincronofasor (Corrente e Tensões) para aquisitar dados via SCADA em nível de distribuição, em *Smart Grids* com topologias reconfiguráveis. E propõem a reconfiguração da grid para obter o menor custo considerando possíveis perdas de potência em consumidores.

Obs.: PMUs, d inglês: "*Phasor Measurement Unit*" Unidade de Medição Fasorial / Síncrono Fasor

- A hybrid FCW-EMD and KF-BA-SVM based model for short-term load forecasting [6]

Estuda o carregamento de um transformador em uma subestação no sul da China decompondo sinal dos dados históricos e projeta a carga futura com utilizando a correlação entre o carregamento em diferentes dias da própria serie de dados, a aplica técnica de pesos *Fuzzy combined weigh* e Filtro de Kalman.

- Energy Time Series Forecasting Based on Empirical Mode Decomposition and FRBF-AR Model [7]

Modelo Híbrido Utiliza EMD método de decomposição empírica, obtendo funções de modo intrínsecas (IMFs) e os resíduos aproximados através de FRBF - *Fuzzy Radial Basis Linear* e essa previsão foi comparada aos dados da *Australian Energy Market Operator* (AEMO) onde o erro quadrado médio foi de 1,24% que parece performar bem. Entretanto, utiliza apenas a série histórica para determinar a predição alguns questionamentos de como é o clima no sul da Austrália e se esse modelo se aplica a outras regiões necessitam de uma discussão.

- *A Short-Term Load Forecasting Method Using Integrated CNN and LSTM Network* [8]

Defende a previsão de carregamento como alternativa a expansão da rede e divide o estudo em: *Short-term Load Forecasting* (STLF): Próximas horas até próxima semana; *Medium-term Load Forecasting* (MTLF): Mais de uma semana até alguns meses; e *Long-term Load Forecasting* (LTLF): Alguns meses até próximos anos. Comparando com outros métodos obteve melhores resultados de previsão utilizando modelo hybrido de *long short-term memory networks* (LSTM) associado a *convolutional neural network* (CNN).

Todos estes trabalhos mencionados fazem o uso de alguma técnica de inteligência artificial, para prever em diferentes períodos (longos ou curtos) de consumo de energia, em sua maioria utiliza-se a própria série histórica para definir e para previsão, em alguns poucos se utiliza a medida em tempo real, para corrigi-la. Neste sentido, o estudo aqui realizado se serve do conhecimento construído ao aplicar inteligência artificial, utilizando mais especificamente redes neurais do tipo LSTM e variáveis históricas de consumo de energia, climáticas, econômicas e sociais, para uma previsão médio prazo MTLF, se diferenciando dos trabalhos anteriores ao considerar um evento aleatório de grandes proporções, como a pandemia da COVID, avaliando a interação da saúde, do clima e da economia no consumo de energia elétrica o que é possível com a metodologia aqui proposta.

1.5 Objetivo

O presente trabalho objetiva a construção de um modelo estocástico de projeção de consumo de energia elétrica, através do emprego de inteligência artificial e otimização genética com um índice de erro inferior a 5%, utilizado para estabelecer esta meta a Resolução Normativa nº 414 da ANEEL [9], na presença de evento aleatório de grandes proporções.

1.6 Estrutura do trabalho

Para atingir o objetivo do trabalho, o mesmo está estruturado em quatro capítulos, incluindo este introdutório, descritos a seguir.

O Capítulo 2 apresenta a metodologia proposta, descreve os métodos, as ferramentas e o desenvolvimento utilizados na construção deste trabalho.

O Capítulo 3 descreve a aplicação da metodologia e do modelo num Estudo de Caso BR, com avaliação dos resultados.

O Capítulo 4 aborda as conclusões sobre os resultados obtidos, e também cita brevemente um trabalho a ser publicado, com parte do estudo aqui realizado.

Complementam o trabalho oito Anexos, que tratam de análises detalhadas dos dados em relação as apresentadas ao longo do texto, resultados dos testes realizados, e o código fonte implementado.

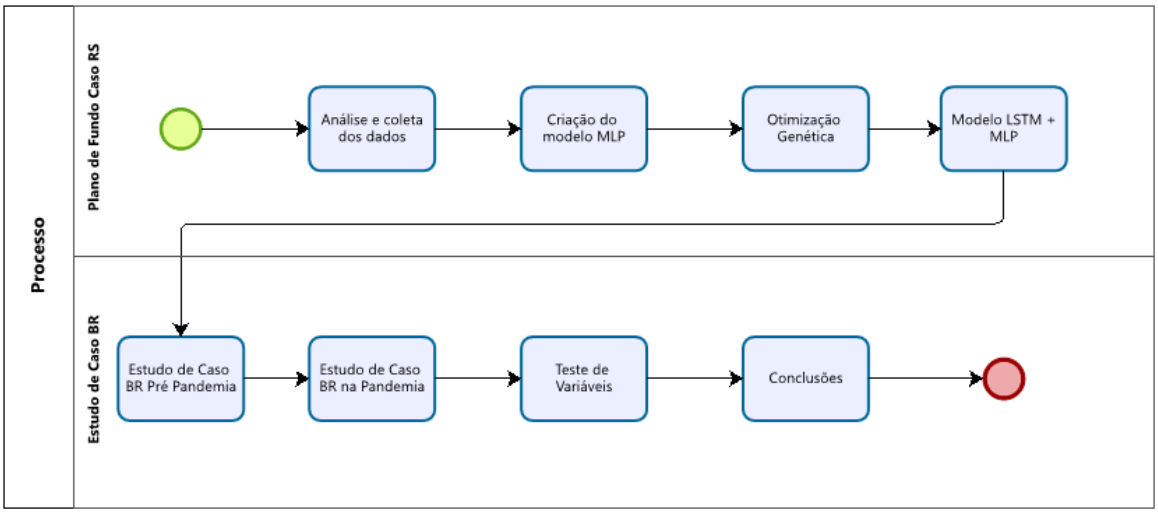
2 METODOLOGIA

Este capítulo apresenta cada etapa em itens e subitens de modo a facilitar a leitura e a repetibilidade deste experimento. Em linhas gerais, o trabalho investiga variáveis correlatas com o consumo de energia elétrica no Brasil, visando à criação de um previsor de consumo de energia. Para isto é realizado um estudo de caso com dados do Rio Grande do Sul, utilizando entradas clássicas como meteorológicas e econômicas, como aspecto de inovação faz uso de dados sociais e de saúde buscando descrever o comportamento anômalo da queda da demanda durante a pandemia de COVID-19, utiliza dados de saúde e/ou de óbitos possam antever o impacto da pandemia na economia. Se essa hipótese estiver correta pode-se evidenciar cientificamente a correlação inversa entre a pandemia e o consumo, onde a causa da aceleração da pandemia significa que o efeito é o freio do consumo/demanda.

Esse estudo é realizado privilegiando e ao mesmo tempo restringindo-se à utilização de dados de entrada de fácil acesso e sem restrição de uso para fins acadêmico, optou-se por dados de institutos governamentais de domínio público. Da mesma forma foram utilizadas ferramentas de código aberto ou sem restrições de uso, em especial a linguagem de programação Python e bibliotecas associadas, que são oportunamente descritas.

A Metodologia do trabalho proposto está sintetizada na Figura 1 e detalhada nas subseções a seguir.

Figura 1 – Metodologia do trabalho.



Fonte: Autor

2.1 Coleta de Dados

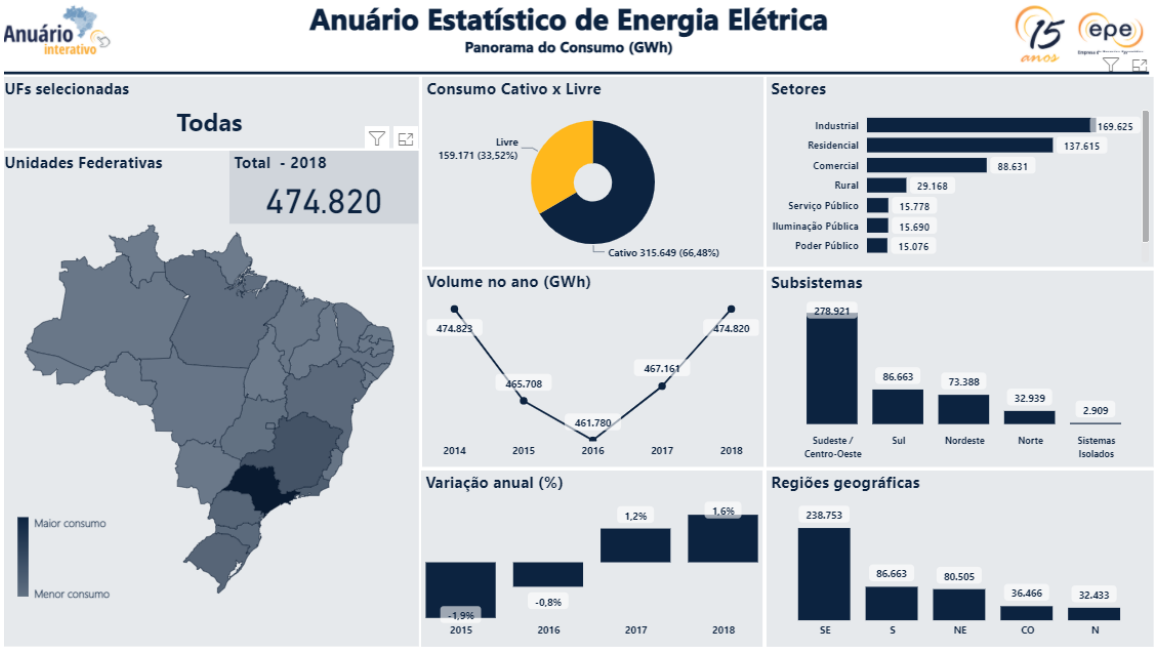
Os dados foram coletados de dados abertos, disponibilizados por institutos governamentais, e assim são de livres utilização para fins acadêmicos.

- EPE - Empresa De Pesquisa Energética

"A Empresa de Pesquisa Energética – EPE tem por finalidade prestar serviços ao Ministério de Minas e Energia (MME) na área de estudos e pesquisas destinadas a subsidiar o planejamento do setor energético, cobrindo energia elétrica, petróleo e gás natural e seus derivados e biocombustíveis. É uma empresa pública federal, dependente do Orçamento Geral da União. A empresa foi criada por meio de medida provisória convertida em lei pelo Congresso Nacional - Lei 10.847, de 15 de março de 2004. E a efetivação se deu em um decreto de agosto de 2004." (Fonte: [10])

Da EPE foram extraídos os dados elétricos do anuário 2019 ano base 2018 [11]. Alguns destes dados podem ser observados na Figura 2, em infográfico extraído do próprio site da EPE.

Figura 2 - Anuário Estatístico de Energia Elétrica.



Fonte: [12]

Posteriormente para avaliar dados mais atuais e o possível efeito da pandemia no consumo de energia, foram utilizados os dados da resenha mensal EPE [13] que são uma prévia do anuário com dados preliminares.

Figura 3 - Recorte da tabela síntese do relatório Resenha Mensal EPE

TABELA SÍNTESE

	EM JUNHO			ATÉ JUNHO			12 MESES		
Consumo (GWh)	2021	2020	%	2021	2020	%	2021	2020	%
SETORES									
BRASIL	40.156	35.696	12,5	249.954	232.139	7,7	493.464	471.650	4,6
RESIDENCIAL	11.956	11.396	4,9	76.787	73.194	4,9	151.766	143.110	6,0
INDUSTRIAL	14.978	12.541	19,4	89.760	78.531	14,3	177.561	162.826	9,0
COMERCIAL	6.678	5.611	19,0	43.425	41.649	4,3	84.300	86.652	-2,7
OUTROS	6.543	6.147	6,4	39.983	38.765	3,1	79.836	79.061	1,0
SUBSISTEMAS									
SISTEMAS ISOLADOS	234	223	5,1	1.443	1.419	1,7	2.972	2.905	2,3
NORTE	3.138	2.827	11,0	18.075	16.833	7,4	36.649	34.833	5,2
NORDESTE	6.349	5.594	13,5	39.055	36.188	7,9	76.298	74.055	3,0
SUDESTE/C.OESTE	23.108	20.549	12,5	144.159	133.718	7,8	286.398	272.658	5,0
SUL	7.327	6.503	12,7	47.221	43.980	7,4	91.147	87.200	4,5

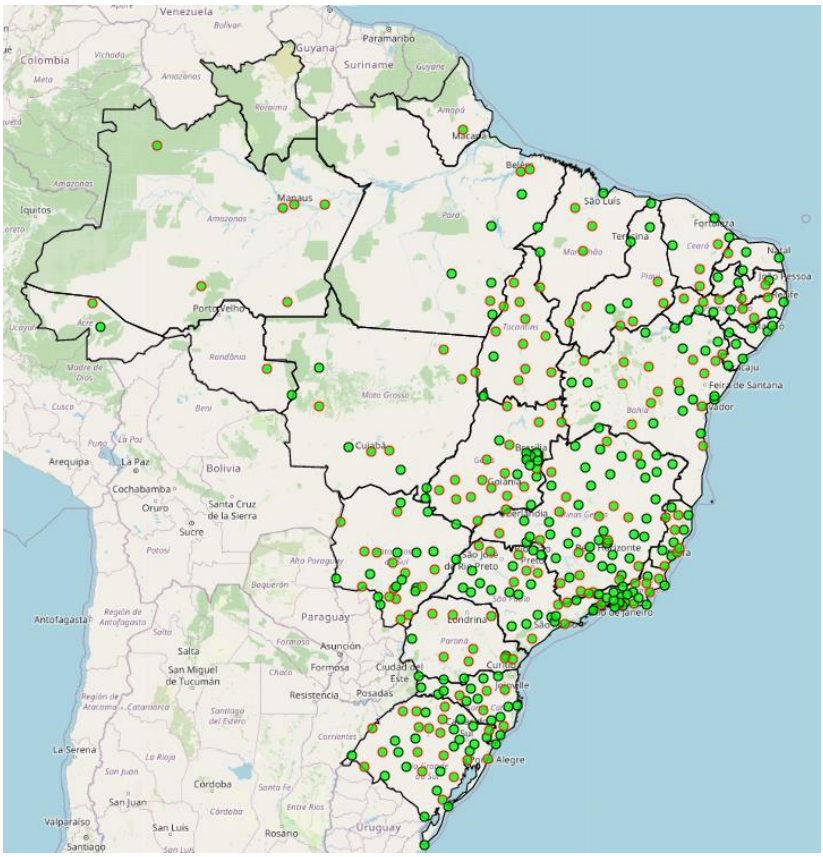
Fonte: [13]

- INMET - Instituto Nacional de Metrologia

> *"O Instituto Nacional de Meteorologia do Brasil (INMET) é um órgão federal da administração direta do Ministério da Agricultura, Pecuária e Abastecimento (MAPA), criado em 1909 com a missão de prover informações meteorológicas através de monitoramento, análise e previsão do tempo e clima, concorrendo com processos de pesquisa aplicada para prover informações adequadas em situações diversas, como no caso de desastres naturais como inundações e secas extremas que afetam, limitam ou interferem nas atividades cotidianas da sociedade brasileira." [14]*

Do INMET foram extraídos os dados meteorológicos disponíveis de todas as estações automatizadas do Rio Grande do Sul, para o estudo de caso do RS, e após utilizou-se os dados das demais estações para o estudo de caso do Brasil. No mapa da Figura 4 são destacadas as estações em todo o Brasil.

Figura 4 - Mapa de Estações INMET Automáticas



Fonte: [14]

- DataSUS/tabnet

"Após a criação da Lei de Acesso à Informação, todas as informações produzidas ou custodiadas pelo poder público são públicas, e portanto, disponíveis a todos os cidadãos, exceto aquelas que são sigilosas por lei" [15]. Entretanto, mesmo disponíveis na maioria das vezes não estão em um formato acessível ou amigável a uma análise de dados, sejam por estar em formatos de imagem ou arquivos formato PDF, não estruturados, ou por estarem pulverizados em vários sub arquivos. Pretendia-se inicialmente utilizar os dados de óbitos do Portal Transparência Registro Civil [16], mas os mesmos não estavam disponíveis para o mesmo período deste estudo, então a alternativa encontrada foi a de utilizar os dados do SUS da plataforma DataSUS/Tabnet [17], que apesar de não ser o dado oficial consolidado pelo Registro Civil, é uma amostra suficientemente representativa e confiável para que possa ser utilizada.

- BCB - Banco Central do Brasil

O Banco Central do Brasil (também conhecido por BC, BACEN ou BCB) é uma autarquia federal integrante do Sistema Financeiro Nacional, iniciou suas atividades em março de 1965. O BC é uma das principais autoridades monetárias do país e tem autonomia para atuar em busca da estabilidade de preços, zelar pela estabilidade e pela eficiência do sistema financeiro, suavizar as flutuações do nível de atividade econômica e fomentar o pleno emprego.

Do BCB [18] foram coletados os indicadores IBC-BR e IPCA. O IBC-BR é o índice que mede a atividade econômica no país e este foi escolhido tendo por base o artigo do próprio BC que avalia intensidade da pandemia e atividade econômica [19]. O IPCA foi selecionado por ser um indicador clássico que mede a inflação no país, e por ser um dos índices divulgados pelo relatório Focus do BC. Esse relatório informa e algumas projeções dos índices economicamente relevantes, além destes indicadores ter o compromisso da equipe econômica de ser mantidos dentro de uma meta de governo pré-definida pela política econômica do país.

- B3 – Brasil Bolsa Balcão

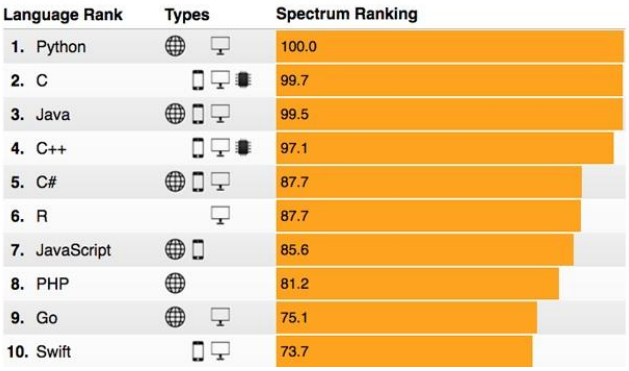
A B3 é a bolsa de valores oficial do Brasil, A B3 surgiu sob o formato atual após a fusão da Bolsa de Valores, Mercadorias e Futuros de São Paulo (BM&FBOVESPA) com a Central de Custódia e de Liquidação Financeira de Títulos (CETIP), aprovada pela Comissão de Valores Mobiliários (CVM).

Da B3 foi utilizado o seu principal índice IBOV – Ibovespa – é o indicador do desempenho médio das cotações das ações negociadas. É formado pelas ações com maior volume negociado nos últimos meses. O valor atual representa a quantia, em moeda corrente, de uma carteira teórica de ações, constituída em 2 de janeiro de 1968, a partir de uma aplicação hipotética [20].

2.2 Ferramentas Utilizadas

Visando manter a solução passível de revisão e garantindo a replicação deste estudo, foi utilizada a linguagem *python* que além de ser uma linguagem de código aberto é a linguagem mais usada nos artigos IEEE, conforme *ranking Spectrum* detalhado na Figura 5.

Figura 5 - Ranking de linguagens mais populares em publicações IEEE



Fonte: [21]

As ferramentas utilizadas estão ilustradas na Figura 6 e são brevemente descritas a seguir.

Figura 6 - Ferramentas Python

Fonte: Autor, logos extraídos das respectivas páginas.

- *Google Colaboratory* [22]

Para a implementação deste trabalho, foi utilizada a plataforma Google Colaboratory como ambiente de desenvolvimento integrado (IDE, do inglês *"Integrated Development Environment"*), que permite rapidamente acessar um *Jupyter Notebook* que é uma implementação *python* que onde é possível mesclar texto e código de forma iterativa e rodar trechos de código de maneira independente, como o google colab. está disponível online não necessita instalação e pode ser acessado de qualquer dispositivo. O que possibilita uma flexibilidade relevante.

- *Matplotlib* [23]

O *Matplotlib* é uma biblioteca *Python* que fornece um grande conjunto de ferramentas para a geração de gráficos dos mais variados tipos e formas, bem como o controle de todos os atributos de impressão.

- *Numpy* [24]

NumPy é uma biblioteca *Python* que suporta *arrays* e matrizes multidimensionais, possuindo uma larga coleção de funções matemáticas para trabalhar com estas estruturas. Foi utilizada em conjunto com as outras ferramentas para manipular e gerar dados, funções e resultados.

- Pandas [25]

Para a manipulação dos dados de forma tabular no formato de *"data frames"* e permitir operações similares a linguagem SQL (Linguagem de Consulta Estruturada), assim como nas análises de correlação e dispersão.

- *Scikit Learn* [26]

O *Scikit Learn* é uma ferramenta voltada ao desenvolvimento de inteligência artificial, entretanto aqui foram utilizadas suas métricas de erro e a regressão linear. Para ranquear as melhores versões de solução.

- *Seaborn* [27]

O *Seaborn* é uma biblioteca que a facilita a geração de gráficos, através de comandos simples tem uma saída complexa sintetizando em poucas linhas os recursos da *matplotlib*.

- *Statsmodels* [28]

O *Statsmodel* é um pacote de análise estatística e deste foram utilizadas as decomposições de sinal afim de verificar a periodicidade e tendência dos dados analisados.

- *Tensor Flow / Keras* [29]

O *TensorFlow* é uma plataforma completa de código aberto para *machine learning*. Ele tem um ecossistema abrangente e flexível de ferramentas, bibliotecas e recursos. Do *tensor flow* mais especificamente foram utilizadas as APIs (*Application Programming Interface*) do Keras que é um subconjunto de ferramentas do *Tensor Flow*.

- Versões utilizadas

Para o desenvolvimento do modelo proposto foram utilizadas as versões descritas na Tabela 1.

Tabela 1 - Versões Utilizadas

Recurso	Versão
matplotlib	3.2.2
numpy	1.19.5
pandas	1.1.5
Python	3.7.10
seaborn	0.11.1
sklearn	0.24.1
statsmodels	0.10.2
TensorFlow	2.5.0

Fonte: Autor

2.3 Análise Exploratória dos Dados

Para verificar as possibilidades de análises, que as bases permitem, foi realizada uma análise exploratória dos dados.

Tabela 2 - Dados Anuário EPE (Amostra)

Data	Tipo Consumidor	Sistema	UF	SetorN1	(...)	Tensão N1	(...) 2	Tensão N4	Faixa de Consumo N1	Faixa de Consumo N2	Consumidores	Consumo
1/4/2014	Cativo	Sudeste / Centro-Oeste	ES	Residencial		B - Baixa Tensão		Baixa Tensão (Ex-Iluminação Pública)	Baixa Renda	> 200 kWh	43697.0	4.000
1/12/2015	Cativo	Sudeste / Centro-Oeste	SP	Poder Público		A - Alta Tensão		A-4 - 2,3 a 25 kV	Não Aplicável	Não Aplicável	32.0	353.000
1/6/2013	Cativo	Sul	RS	Poder Público		A - Alta Tensão		AS - < 13,8 kV (Subterrâneo)	Não Aplicável	Não Aplicável	1.0	10.000
1/12/2017	Cativo	Nordeste	PE	Poder Público		A - Alta Tensão		A-4 - 2,3 a 25 kV	Não Aplicável	Não Aplicável	230.0	10.788.000
1/8/2018	Livre	Sudeste / Centro-Oeste	RJ	Comercial		A - Alta Tensão		AS - < 13,8 kV (Subterrâneo)	Não Aplicável	Não Aplicável	21.0	2.009.853

Fonte: [11]

- Análise dos dados Elétricos EPE

Os dados fornecidos pelo anuário estatístico estavam em formato .csv (*Comma-separated values*) o que facilitou sua importação e manipulação utilizando pandas.

Nele pode-se verificar que os dados fornecidos estavam agregados por classes de consumo e níveis de tensão para cada mês de 2013 a 2018, e classificados em região e unidade federativa.

Para o último mês medido há um total de 89.978.322 unidades consumidoras. Subdivididas em setores por até 2 níveis de classificação:

- Comercial
    - Adm. Condominial, Iluminação, Uso Comum
    - Associação e Entidades Filantrópicas
    - Comercial
    - Iluminação em Rodovias por Concessão
    - Outros Serviços e Atividades
    - Semáforos, Radares e Câmeras de Trânsito
    - Serviços de Comunicações e Telecomunicações
    - Serviços de Transporte Exclusive Tração Animal
    - Templos Religiosos
- Consumo Próprio
    - Estação de recarga de veículos elétricos
    - Outras Atividades
- Iluminação Pública
- Industrial
- Poder Público
    - Estadual ou Distrital

    - Federal
    - Municipal
- Residencial
    - Baixa Renda
    - Convencional (Exceto Baixa Renda)
- Rural
    - Agroindústria
    - Agropecuária
    - Agropecuária Urbana
    - Aquicultura
    - Cooperativa de Eletrificação Rural
    - Escola Agrotécnica
    - Rural Residencial
    - Serviço Público de Irrigação
- Serviço Público
    - Água, Esgoto e Saneamento
    - Tração Elétrica

Em faixas de tensão e 2 níveis de consumo para consumidores de tensão N1 Classe B (Baixa tensão):

- A - Alta Tensão
    - A-1 - 230 kV ou superior
    - A-2 - 88 a 138 kV
    - A-3 - 69 kV
    - A-3a - 30 a 44 kV

- A-4 - 2,3 a 25 kV
- Alta Tensão Residencial
- AS - < 13,8 kV (Subterrâneo)

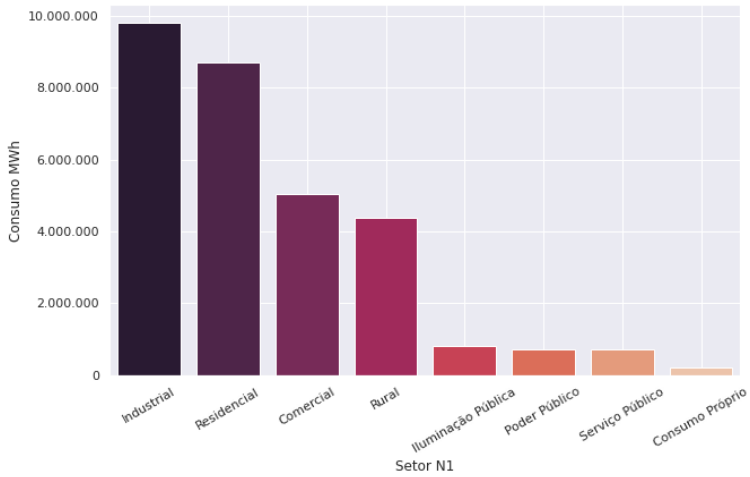
- B - Baixa Tensão
  - B-4
    - B4A Rede de Distribuição
    - B4B Bulbo da Lâmpada
    - Nível de IP Acima do Padrão
  - Baixa Tensão (Ex-Iluminação Pública)
    - Baixa Renda
      - 0-30 kWh
      - 31-100 kWh
      - 101-200 kWh
      - > 200 kWh
    - Convencional
      - 0-30 kWh
      - 101-200 kWh
      - 201-300 kWh
      - 301-400 kWh
      - 31-100 kWh
      - 401-500 kWh
      - 501-1000 kWh

- > 1000 kWh

Para uma visão detalhada foi realizado um recorte nos dados do estado do Rio Grande do Sul de forma a aprofundar o problema em um estudo de caso a um custo computacional menor antes de expandir o método a todos os estados da federação e se conseguir tratar de maneira satisfatória um dado menor.

Visualizando os dados de 2018 do RS pelo primeiro nível de agrupamento EPE SetorN1 visualizando os dados, verifica-se na Figura 7 que os grupos mais significativos são: Industrial, Residencial, Comercial e Rural.

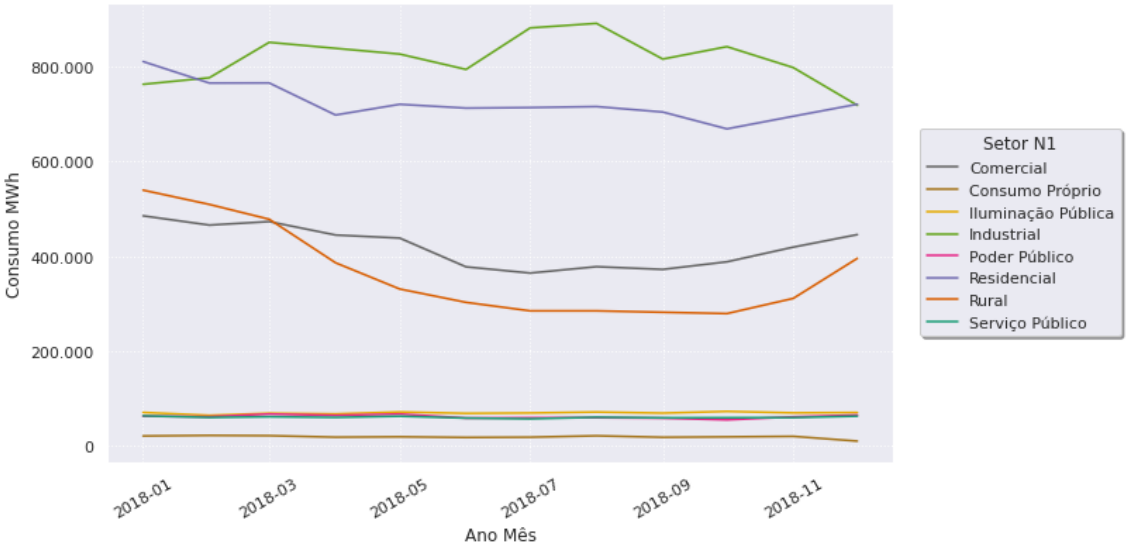
Figura 7 - Consumo por Setor N1 RS em 2018



Fonte: Autor

Já ao analisar a distribuição deste consumo ao logo dos meses, pode-se notar quais são as cargas que apresentam uma variação ao longo do ano. A Figura 8 mostra que o setor Rural apresenta a maior variação ao longo do ano.

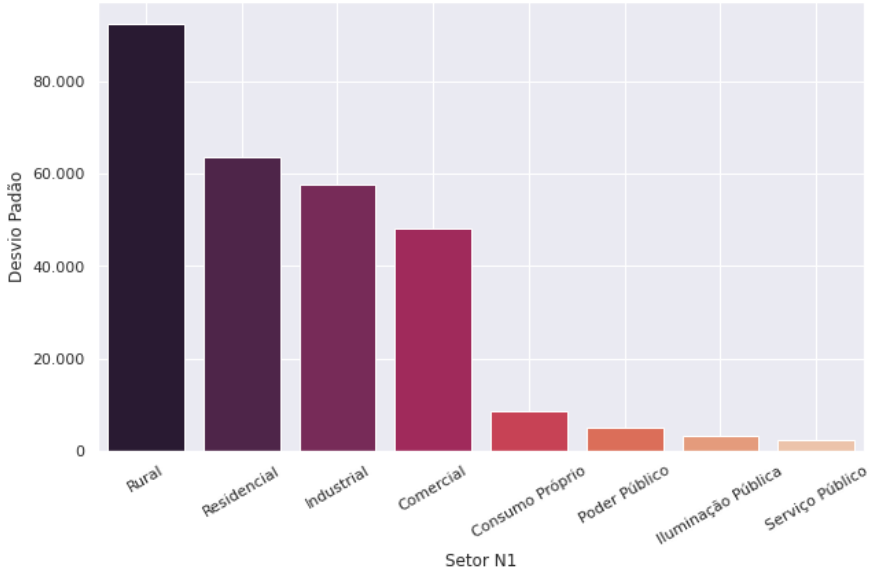
Figura 8 - Perfil de Consumo Mensal por Setor N1 RS em 2018



Fonte**:** Autor

Para validar esta informação e verificar que essa oscilação se repete ao longo da série de 2013 à 2018 foi feita uma análise de dispersão através do desvio padrão, onde os resultados estão mostrados na Figura 9.

Figura 9 – Desvio Padrão de consumo por Setor N1 RS de 2013 à 2018

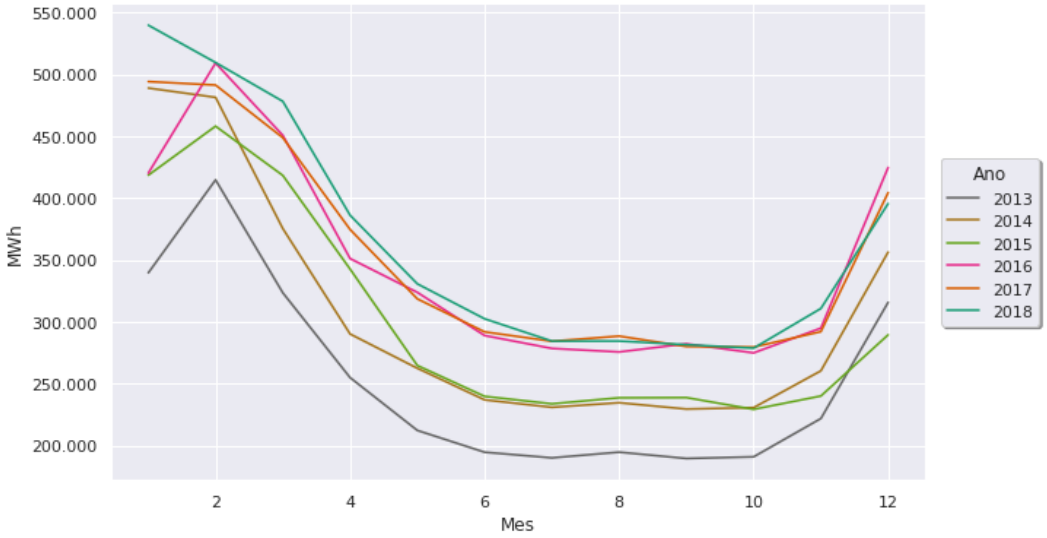


Fonte: Autor

Assim, fazendo um novo recorte e entendendo que o mais desafiador é a previsão de uma carga variável ao longo do tempo, em relação a uma carga constante ou que varie muito suavemente.

Visualizando, então, apenas os dados dos consumidores rurais do RS pode-se aprofundar o estudo nessa classe de consumidores. Na Figura 10 evidencia-se a sazonalidade da carga e percebe-se agora uma tendência de aumento na mesma.

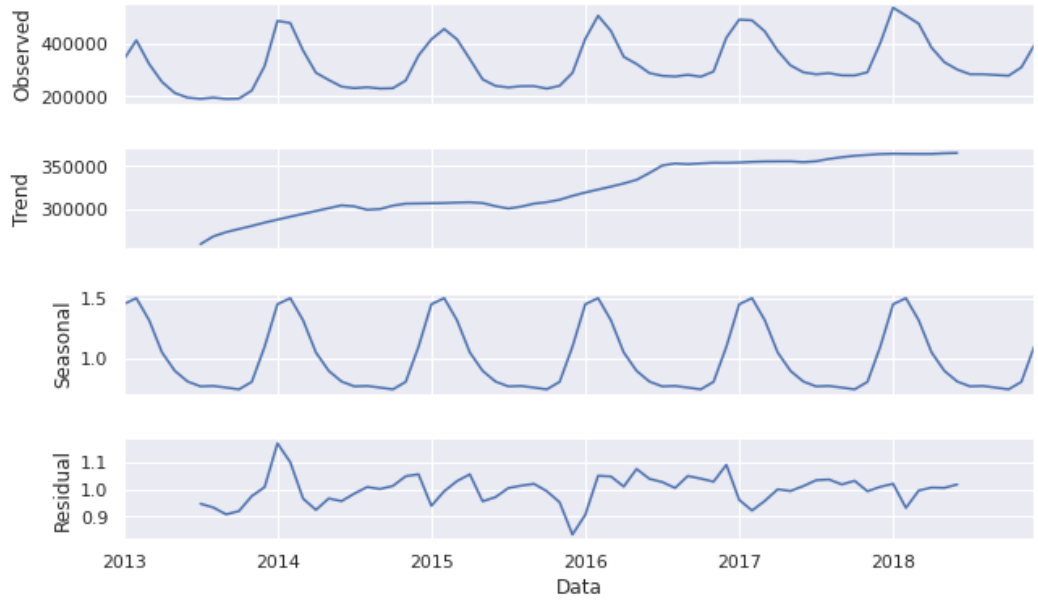
Figura 10 - Perfil de consumo de 2013 a 2018, Detalhe: Setor N1 = Rural e UF = RS



Fonte: Autor

Ao decompor o sinal de consumo ao longo do tempo utilizando a o método multiplicativo do *Statsmodels* [28] na Figura 11 confirma-se a proposição anterior de que se trata de um sinal periódico no tempo e com uma tendência de alta.

Figura 11 - Decomposição Sazonal Multiplicativa do Perfil de Consumo, Detalhe: Setor N1 = Rural e UF = RS

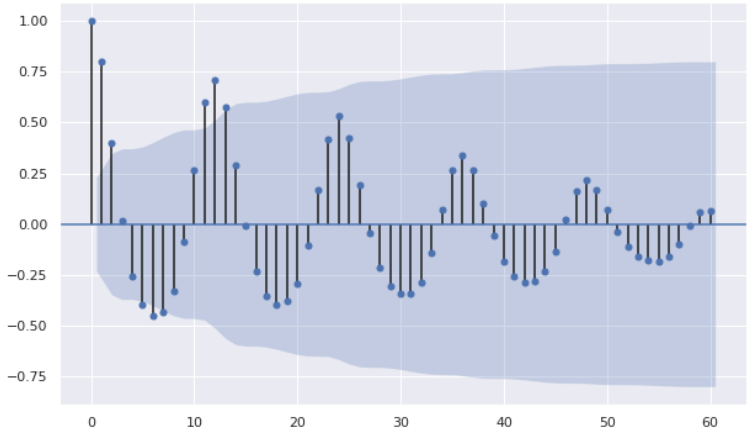


Fonte: Autor

A fim de comprovar-se a sazonalidade e verificar se é possível associar a mesma a sucessão de meses e estações do ano foi aplicado o teste de autocorrelação da série de consumo

Rural no RS sob ela própria defasando-a mês a mês obtém-se o gráfico da Figura 12 onde percebe-se uma correlação negativa após uma defasagem de 4 meses o que enseja uma possível dependência das estações do ano. A característica senoidal reforça essa hipótese e a envoltória decrescente reafirma a tendência de incremento no consumo com o passar dos anos.

Figura 12 – Autocorrelação, Detalhe: Setor N1 = Rural e UF = RS



Fonte: Autor

Ainda foi realizado uma breve análise sobre os outros setores N1 onde se pode verificar seguindo os mesmos critérios uma sazonalidade clara nos setores N1: Residencial Comercial, Rural, Poder Público. Um comportamento não diretamente correlato a sucessão dos meses do ano, nos setores N1: Industrial, Iluminação Pública, Serviço Público, Consumo Próprio, melhores descritos no Anexo 1 – Análise de Consumo Mensal por Setor N1 no RS.

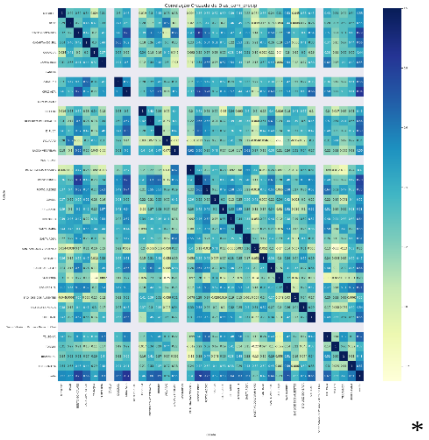
- Análise dos Dados Meteorológicos do INMET

Devido á característica sazonal verificada nos dados elétricos e da atividade rural depender essencialmente do clima e das estações, buscou-se os dados do INMET e desta base de dados do foram selecionadas as estações de coleta automáticas (A distribuição geográfica destas estações pode ser verificada na Figura 4) nestas estações estavam disponíveis os seguintes dados agrupados por mês:

- Número de dias com precipitação
- Precipitação total, mensal [mm]
- Pressão Atmosférica, média mensal [mbar]
- Temperatura média, mensal [°C]
- Vento, velocidade máxima mensal [m/s]
- Vento, velocidade média mensal [m/s]

Para verificar quais destes dados seriam relevantes e se seria possível tomar os dados de uma estação como representação do todo foi realizada uma análise de correlação de Pearson para cada uma destas variáveis comparando sempre estre todas as estações e com o valor médio de todas elas para cada medida. Obtendo ao avaliar dados INMET do RS de 2013 a 2018, os mapas de correlação cruzada conforme exemplo da Figura 13, os mapas completos podem ser consultados no Anexo 2 – Mapas de Correlação Meteorológicos.

Figura 13 - Exemplo de Mapa de Correlação Cruzada

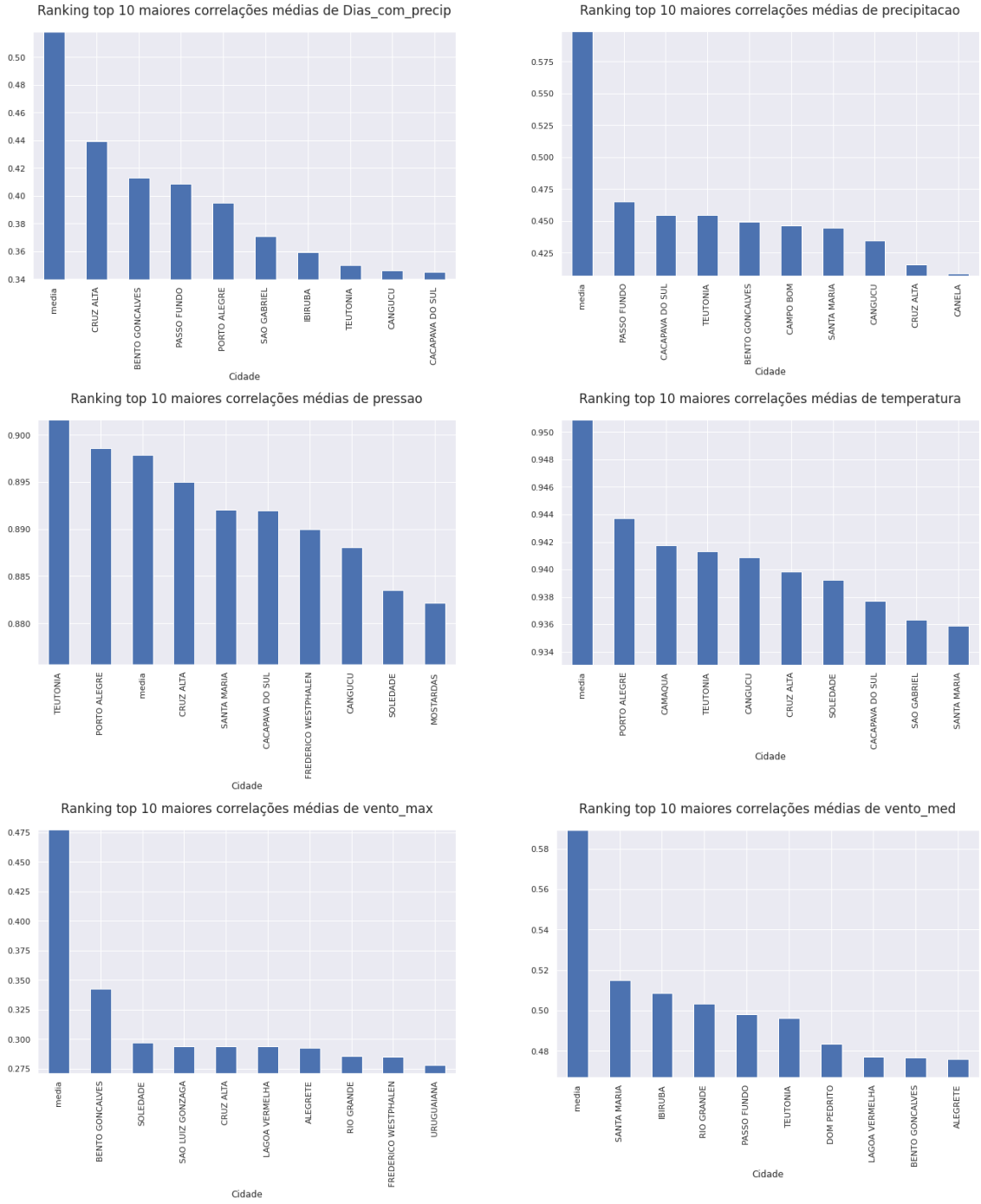


Fonte: Autor

Para cada uma destas variáveis foi tomada a média das correlações e em seguida ranqueadas em ordem decrescente para verificar quais seriam as estações mais representativas Figura 14.

*Obs.: As figuras destes mapas (exemplo da Figura 13) foram descritas em anexo para possibilitar ao leitor as leia em tamanho apropriado.

Figura 14 - Ranking das Maiores Médias de Correlação Entre Estações Automáticas INMET Por Variável meteorológica



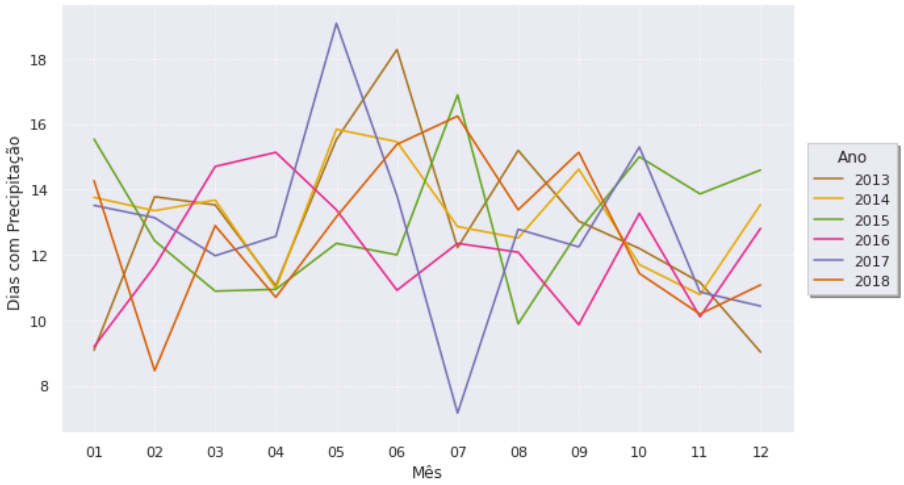
Fonte: Autor

Verificou-se que a estação mais significativa não é nenhuma delas e sim a média de todas, que teve maior correlação média em 5 dentre as 6 variáveis avaliadas.

Seguindo o estudo com a utilização das médias de cada variável, se avaliou primeiramente o comportamento ao longo dos meses, buscando uma correlação direta ou inversa com o caso estudado, do consumo de energia Rural no RS

Na Figura 15 não se observou um padrão bem definido no número médio de dias com precipitação.

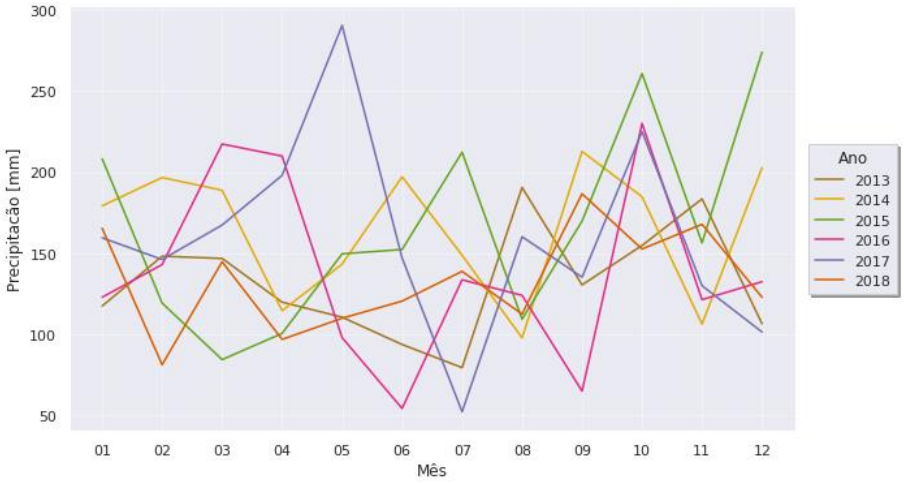
Figura 15 - Perfil Médio de Dias com Precipitação no RS



Fonte: Autor

Da mesma forma na Figura 16 não é nítido um padrão bem estabelecido para a precipitação acumulada.

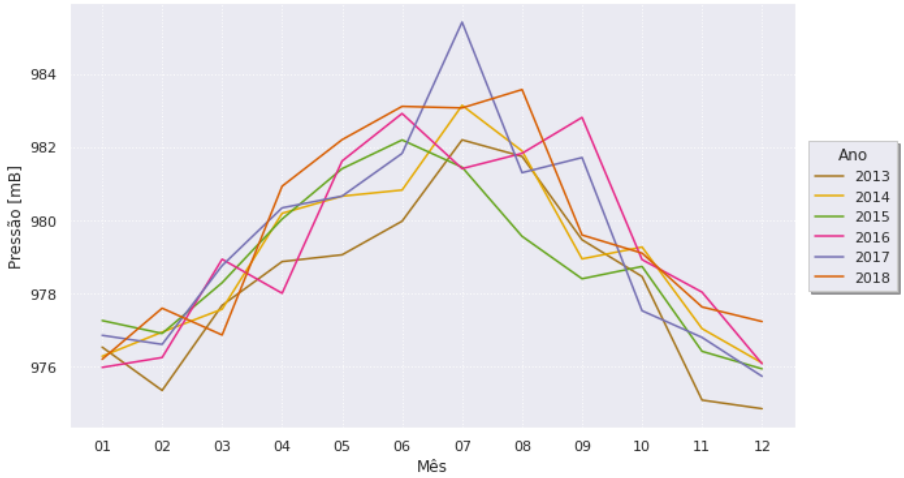
Figura 16 - Perfil Médio de Precipitação Acumulada no RS



Fonte: Autor

Na Figura 17 verifica-se um padrão de pressão ao longo do ano que se repete ao longo dos anos para a série avaliada.

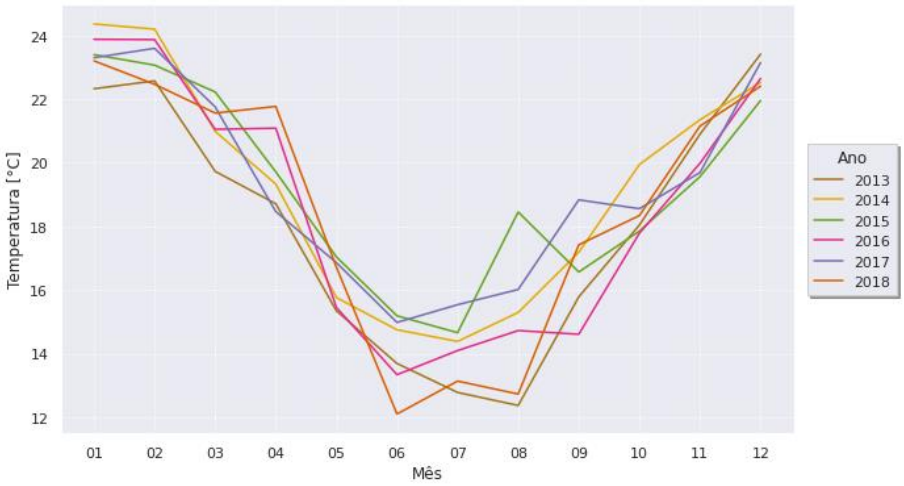
Figura 17 - Perfil Médio de Pressão no RS



Fonte: Autor

Para o perfil de temperatura na Figura 18 também se verifica um padrão bem estabelecido.

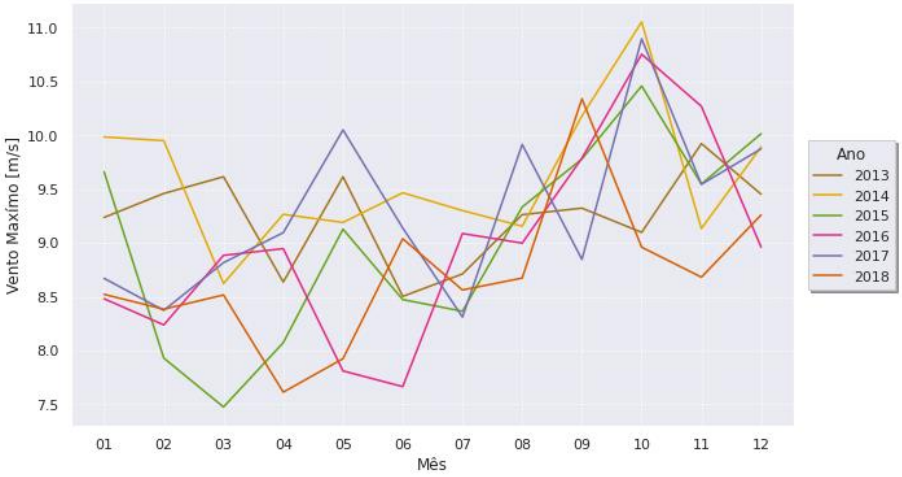
Figura 18 - Perfil Médio de Temperatura no RS



Fonte: Autor

Na Figura 19, referente ao perfil dos ventos máximos é verificado um padrão mais discreto, mas presente com um pico em outubro.

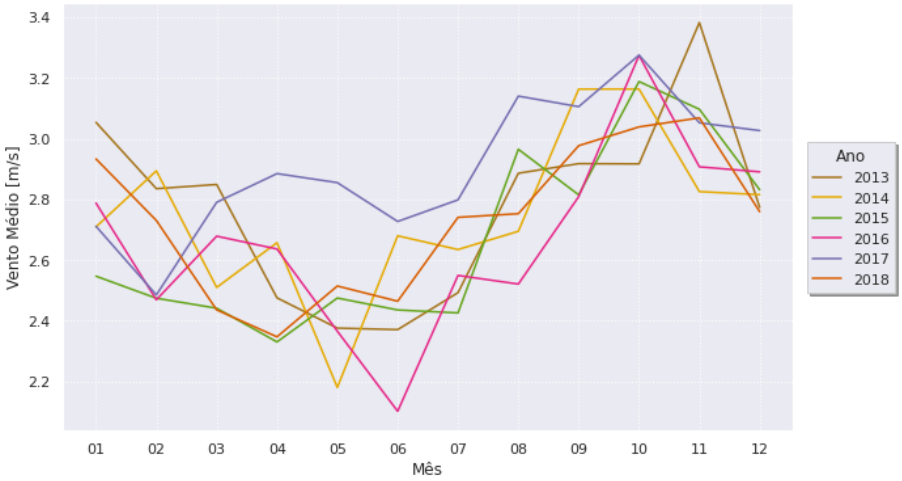
Figura 19 - Perfil Médio de Velocidade Máxima do Vento no RS



Fonte: Autor

A média da velocidade dos ventos mostrada na Figura 20 segue um padrão parecido como era de se esperar, mas mais bem definidos como um padrão que se repete onde a alta de 2016 poderia ser justificada pelo *el niño* de intensidade forte registrado naquele ano conforme [30].

Figura 20 - Perfil Médio de Velocidade Máxima do Vento no RS



Fonte: Autor

Após verificar-se quais variáveis tem um comportamento bem definido nota-se que os principais candidatos seriam a Temperatura, a Pressão e o Vento.

- Análise dos Pandêmicos do Tabnet Sus

Segundo o a ONG Médicos sem fronteiras [31]: “O COVID-19 é a doença provocada pelo novo coronavirus. Mais de 200 países relataram casos da doença e a Organização Mundial

de Saúde (OMS) declarou o surto como uma pandemia, que é uma epidemia que ganha escala global.”(...)

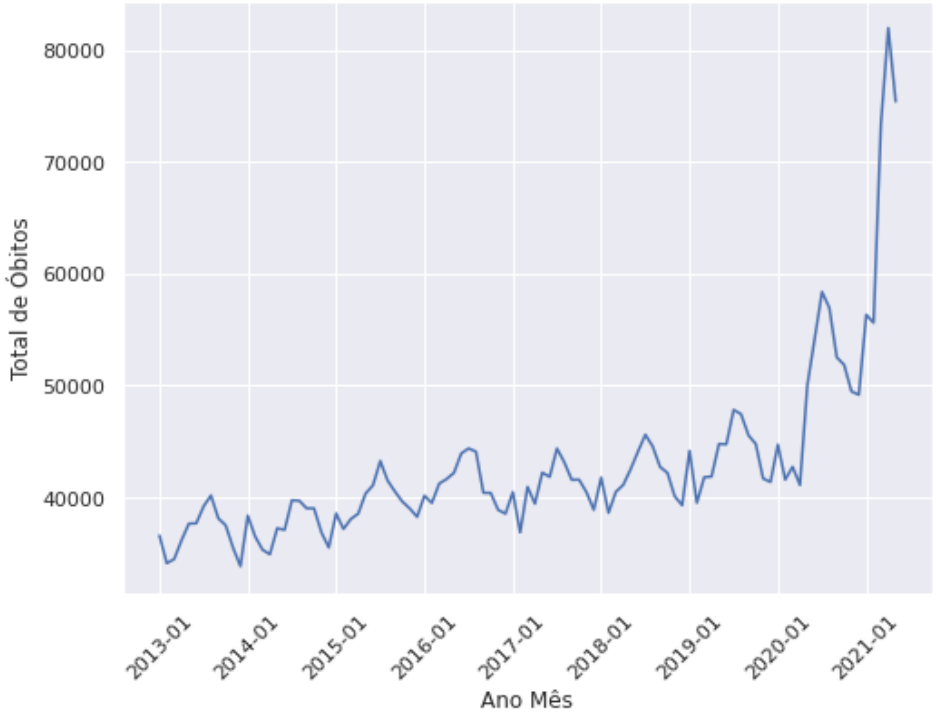
Como no início deste trabalho pouco se sabia sobre os efeitos que uma paralização pandêmica poderia ser capaz de produzir na economia e por conseguinte no consumo de energia elétrica, este trabalho propõe-se a verificar se há evidencias diretas deste possível impacto.

Devido à constatação publicada pelo EPICOVID (Epidemiologia Da Covid-19) [32] de que no Brasil houve até então uma subnotificação dos casos de COVID optou-se por usar os dados de óbitos gerados tanto por SRAG (Síndrome Respiratória Aguda Grave) quanto os dados de COVID como indicadores da gravidade da pandemia no Brasil.

Corroborando essa escolha, o relatório divulgado pelo Banco Central do Brasil – BCB [19], correlaciona o índice de atividade econômica IBC-BR com os índices de mortalidade por COVID ou SRAG constatando que: “Os resultados da análise sugerem a existência de uma relação contemporânea negativa entre a crise sanitária e o nível de atividade econômica. .Contudo, essa relação não é constante ao longo do tempo”(...) "indicando que um agravamento da crise sanitária está associado a deterioração da atividade econômica”. O mesmo relatório pondera que uma atenuação no agravamento econômico apesar do agravamento da COVID pode ser correlacionado com o aumento mobilidade e assim favorecendo as atividades econômicas.

Na Figura 21 – pode-se verificar o comportamento das mortes ao longo dos anos e assim pode-se verificar que o aumento percebido ao longo de dos anos de 2019 e 2020 causados direta ou indiretamente pela pandemia.

Figura 21 – Total de óbitos

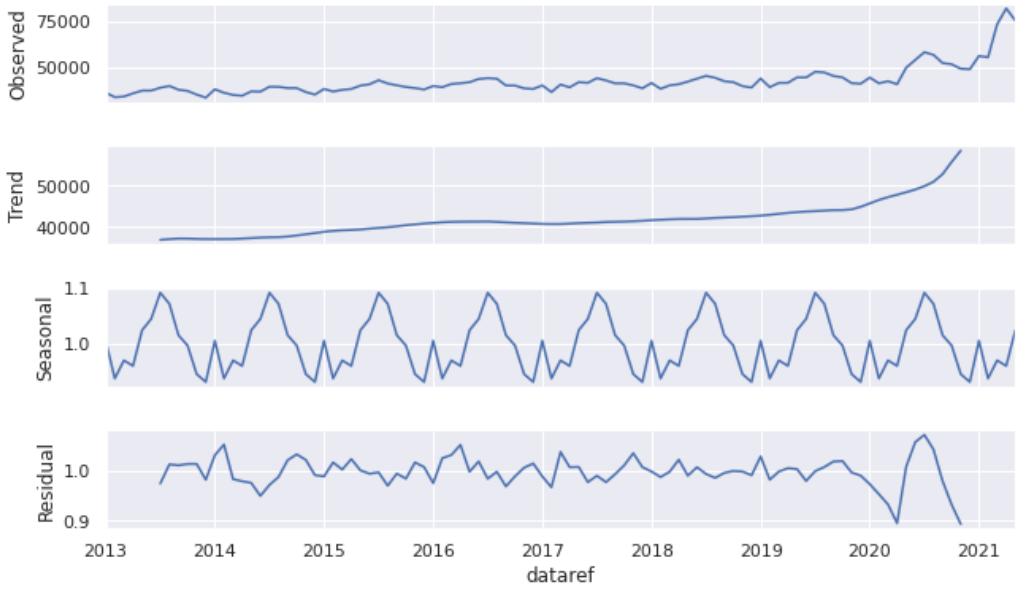


Fonte: Autor

É evidente a quebra do padrão anterior nos anos de 2020 e 2021. Estes dados são oportunamente usados na segunda etapa de modelagem, onde o enfoque é dado para a generalização do modelo e avaliação de novas variáveis.

Essa quebra de tendência e ruptura de padrão da própria série fica mais evidente ao analisar-se a decomposição da série temporal na Figura 22.

Figura 22 - Decomposição sazonal: Óbitos



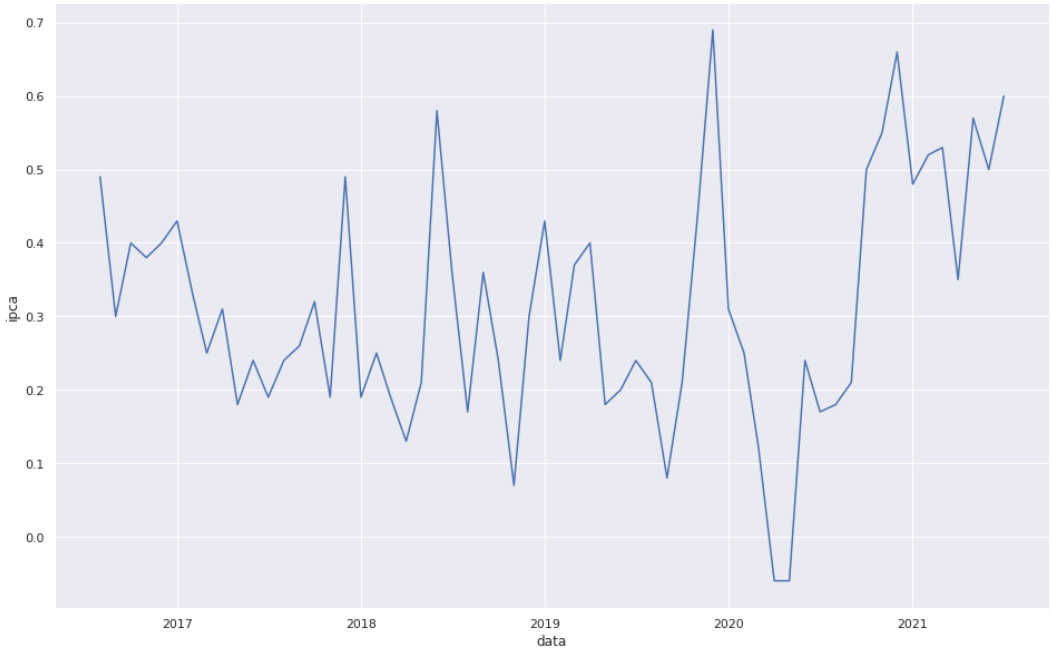
Fonte: Autor

Estes dados serão utilizados na segunda etapa de modelagem onde o enfoque dado para generalização do modelo e avaliação de novas variáveis.

- Análise dos Dados de atividade econômica fornecidos pelo Banco Central do Brasil

Para avaliar a atividade econômica utilizou-se o indicador oficial do IBC-BR, ilustrado na Figura 23.

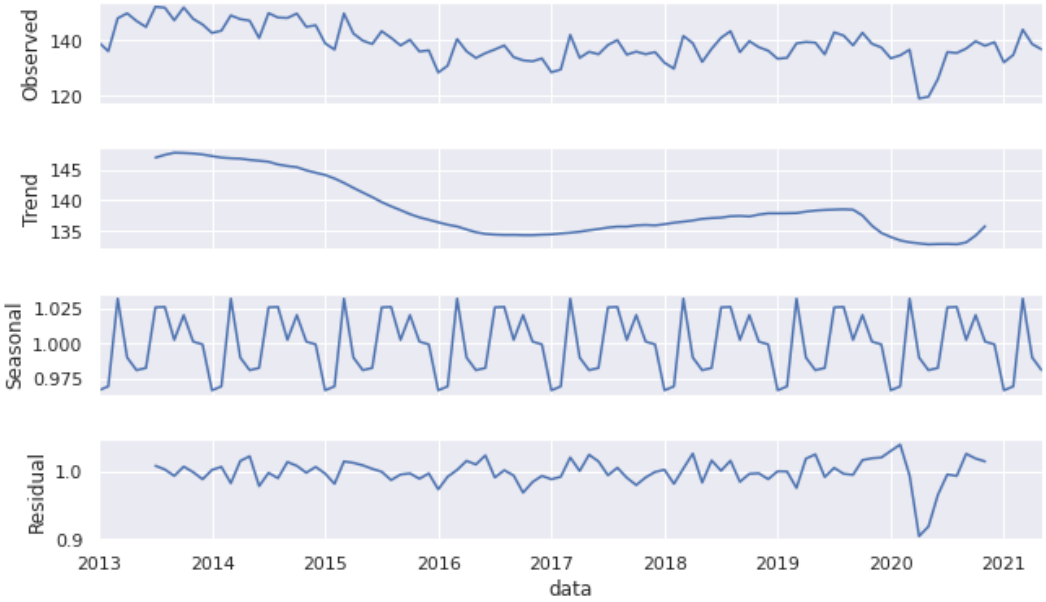
Figura 23 – Perfil de Atividade Econômica no Brasil (IBC-BR)



Fonte: Autor

Novamente verifica-se uma a quebra do padrão anterior nos anos de 2020 e 2021. Com um abrupto degrau em 2020. Essa quebra de tendência e ruptura de padrão da própria série fica mais evidente ao analisar-se a decomposição da série temporal na Figura 24.

Figura 24 - Decomposição sazonal: IBCBR



Fonte: Autor

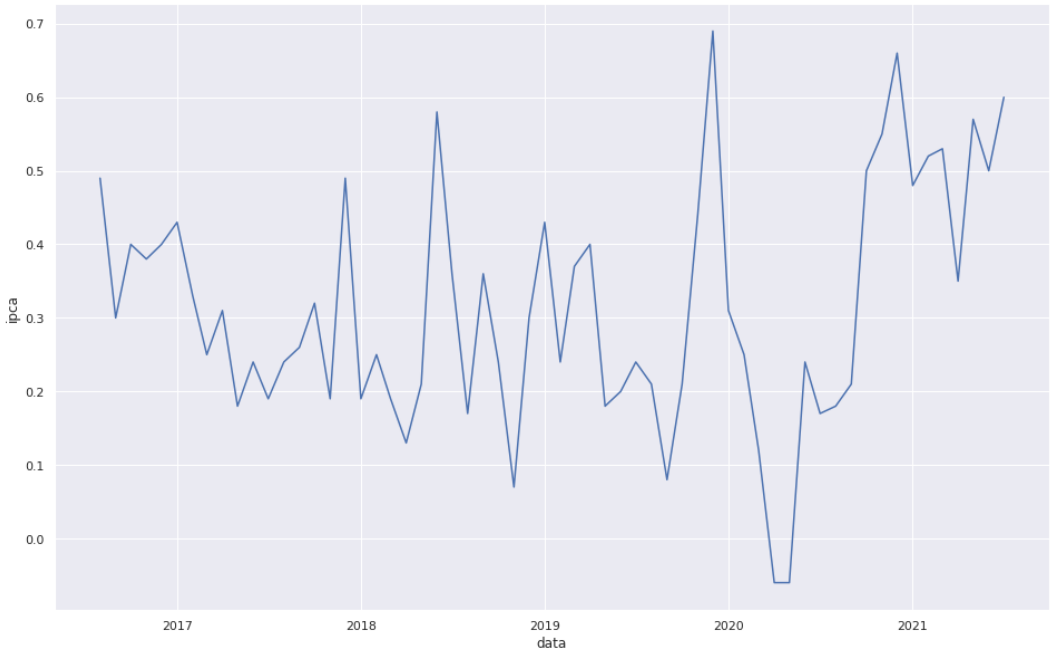
Estes dados serão utilizados na segunda etapa de modelagem onde o enfoque dado para generalização do modelo e avaliação de novas variáveis.

- Análise da série Histórica de atividade econômica IPCA

O IPCA além de um indicador clássico se difere dos anteriores por refletir o custo de vida para famílias com renda mensal de 1 a 40 salários-mínimos. De forma que ao avaliar o custo de vida dessas famílias poder-se-á ter a influência destes na cadeia de consumo e assim essa alteração no poder de compra terá reflexos no consumo de energia elétrica da própria família como também, na indústria ou no comércio e nos fornecedores primários dos produtos consumidos ou não por estas famílias.

Ao observar a Figura 26, não é possível em uma análise preliminar um padrão bem definido.

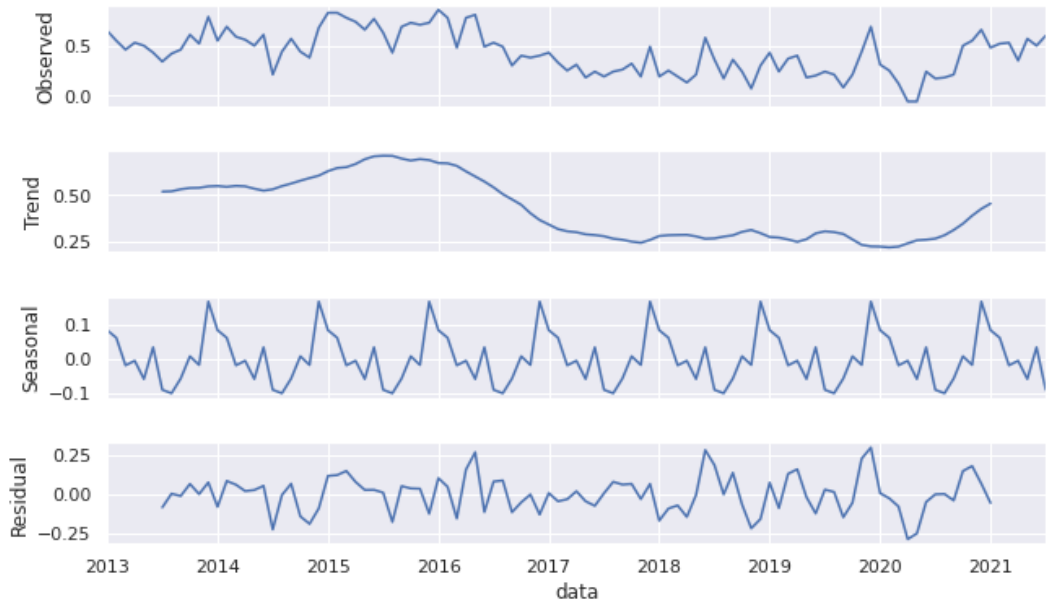
Figura 25 – Perfil de Índice de Preços ao Consumidor Amplo (IPCA)



Fonte: Autor

Já na Figura 26, verificamos que uma tendencia de alta durante a pandemia, a partir do início de 2020. Após um período de estabilidade.

Figura 26 - Decomposição sazonal da série IPCA



Fonte: Autor

- Análise da série histórica IBOV

O índice iBovespa mede a atividade dos negócios na bolsa de valores do Brasil através dos papeis com maior volume negociado. E assim é uma medida de “otimismo” ou “pessimismo” do mercado, como os negociantes tentam antecipar o próprio mercado entende-se que a bolsa varie antes do IPCA ou do IBC-BR, que são medidas de passado enquanto a bolsa é uma expectativa e uma aposta de ganho futuro.

Na Figura 27 é possível verificar a queda abrupta em 2020 em decorrência da pandemia

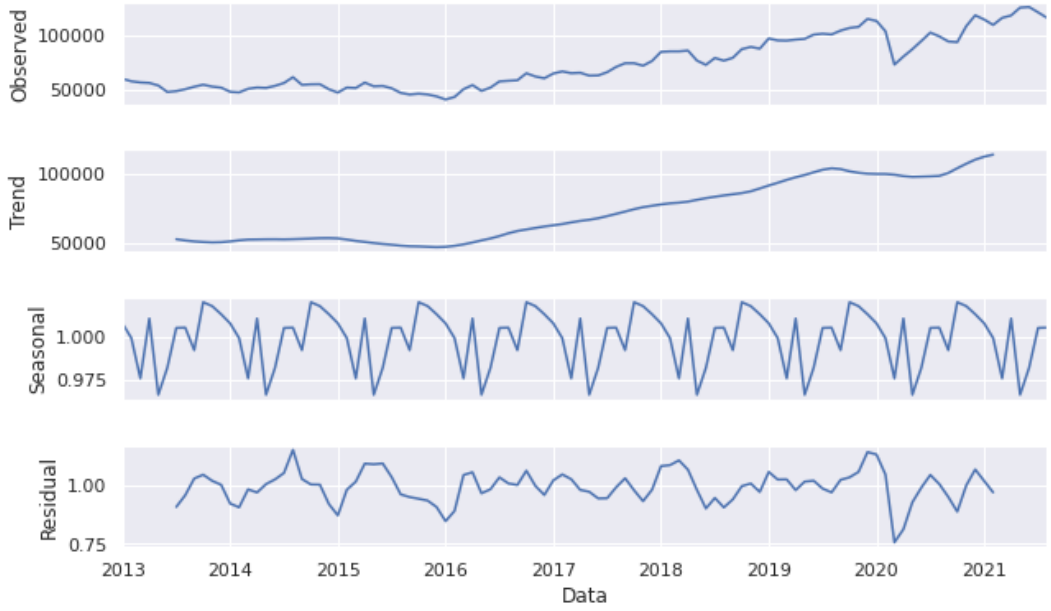
Figura 27 – Perfil de Atividade Econômica no Brasil (IBOV)

Fonte: Autor

Essa quebra de tendência e ruptura de padrão da própria série fica mais evidente ao analisar-se a decomposição da série temporal na Figura 28 ainda verifica-se que a mudança de padrão de tendencia e resíduo ocorre pouco antes que na Essa quebra de tendência e ruptura de padrão da própria série fica mais evidente ao analisar-se a decomposição da série temporal na Figura 24.

Figura 24 do IBCBR por ex. atendendo a expectativa de um sinal preditivo.

Figura 28 - Decomposição sazonal: IBOV



Fonte: Autor

Estes dados serão utilizados na segunda etapa de modelagem onde o enfoque dado para generalização do modelo e avaliação de novas variáveis.

2.4 Análise de Correlação Entre o Consumo vs Clima e Escolha de Variáveis de Entrada

Retomando o estudo de caso do RS, como primeira etapa de modelagem procura-se quais seriam as variáveis que mais influenciam o consumo de energia elétrica rural no Rio Grande do Sul e supõe-se, devido à natureza da atividade agrícola, que as variáveis climáticas seriam as mais representativas. Observa-se uma tendencia de aumento da carga ao logo dos anos também inclui-se nessa etapa a variável ano para dar conta dessa tendencia de alta demonstrada na ao decompor-se o sinal na Figura 11. Então, correlacionado os dados de consumo com os dados climáticos e o ano obtém-se a Figura 29, e nela utilizando os critérios de correlação de Pearson:

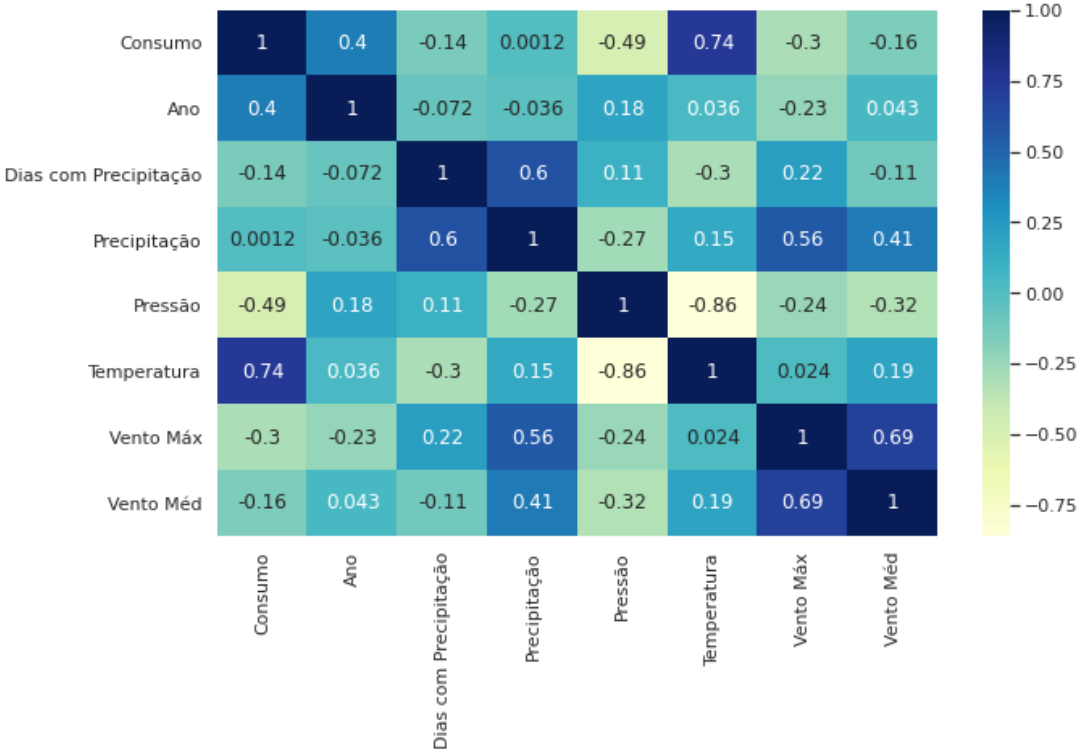
- 0.9 para mais ou para menos indica uma correlação muito forte;
- 0.7 a 0.9 positivo ou negativo indica uma correlação forte;
- 0.5 a 0.7 positivo ou negativo indica uma correlação moderada;
- 0.3 a 0.5 positivo ou negativo indica uma correlação fraca;
- 0 a 0.3 positivo ou negativo indica uma correlação desprezível.

Seleciona-se como variáveis relevantes:

- Ano com correlação fraca de 0,4;
- Pressão Atmosférica, média mensal [mb]: com correlação fraca de -0,49;

- Temperatura média, mensal [°C]: com correlação forte de 0,74;
- Vento, velocidade máxima mensal [m/s] com correlação fraca de - 0,3.

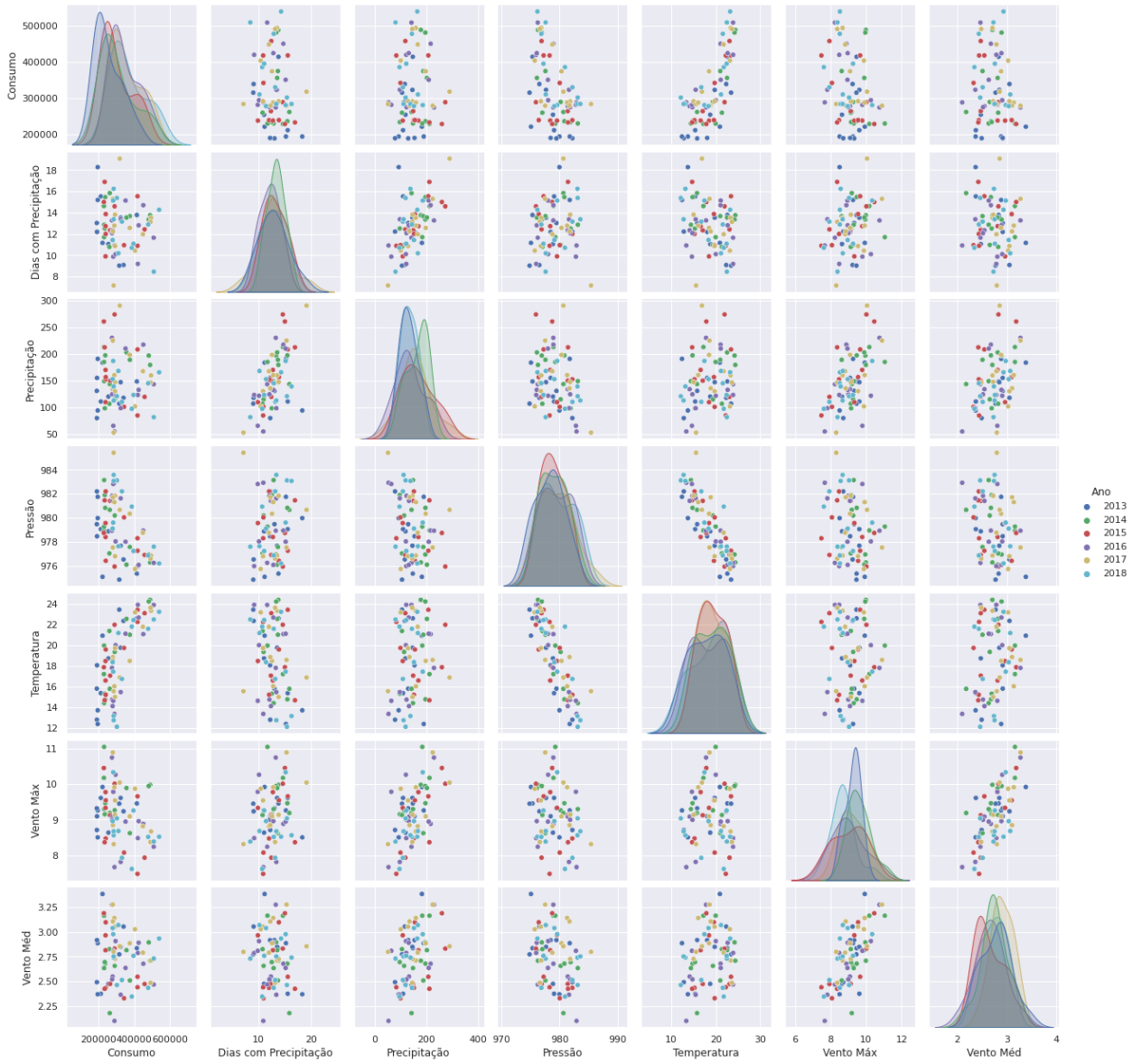
Figura 29 - Correlação entre as variáveis



Fonte: Autor

Na Figura 30 observam-se as distribuições dos dados das mesmas variáveis graficamente em dispersão de pontos e em distribuição normal contrastados por ano que nos traz uma visualização mais sensível do quantificado pela correlação. Ainda, há a possibilidade de se observar na diagonal principal a densidade de distribuição desses dados ao longo dos anos, nota-se um consumo de mesmo perfil, mas com um crescimento, dias com precipitação e pressão com um comportamento similar ao longo dos nos, enquanto os ventos apresentam distribuições diferentes com o passar dos anos.

Figura 30 - Dispersão Cruzada dos Dados e Distribuição Normal



Fonte: Autor

2.5 Previsão de consumo via rede neural artificial

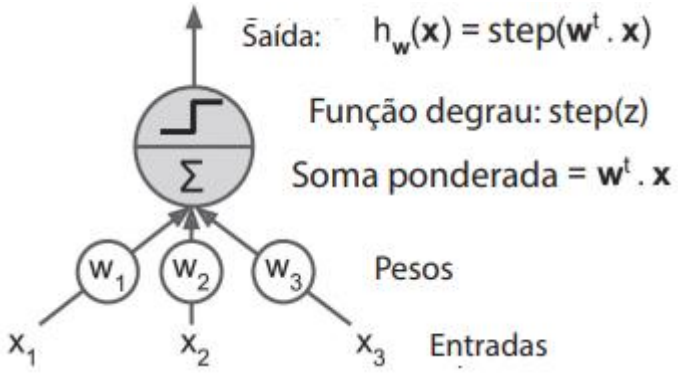
Uma rede neural artificial é um arranjo de células matemáticas computacionais, inspiradas no cérebro humano, e são utilizados como classificadores ou regressores matemáticos que a partir de um conjunto de dados é ajustada para classificar ou estimar um valor com base em dados inéditos.

2.5.1 Um Neurônio *Perceptron*

Um neurônio computacional, possui um determinado número de entradas distintas, estas por sua vez são multiplicadas por uma variável peso individualizado, o produto de peso x entrada é somado a uma constante chamada “bias”, este somatório é o argumento de uma função

de ativação, arbitrária a qual retorna um valor de saída. (Figura 31), este modelo é chamado de *perceptron* o qual teve as primeiras publicações e implementações por Rosenblatt em 1957 [33].

Figura 31 - Neurônio Perceptron



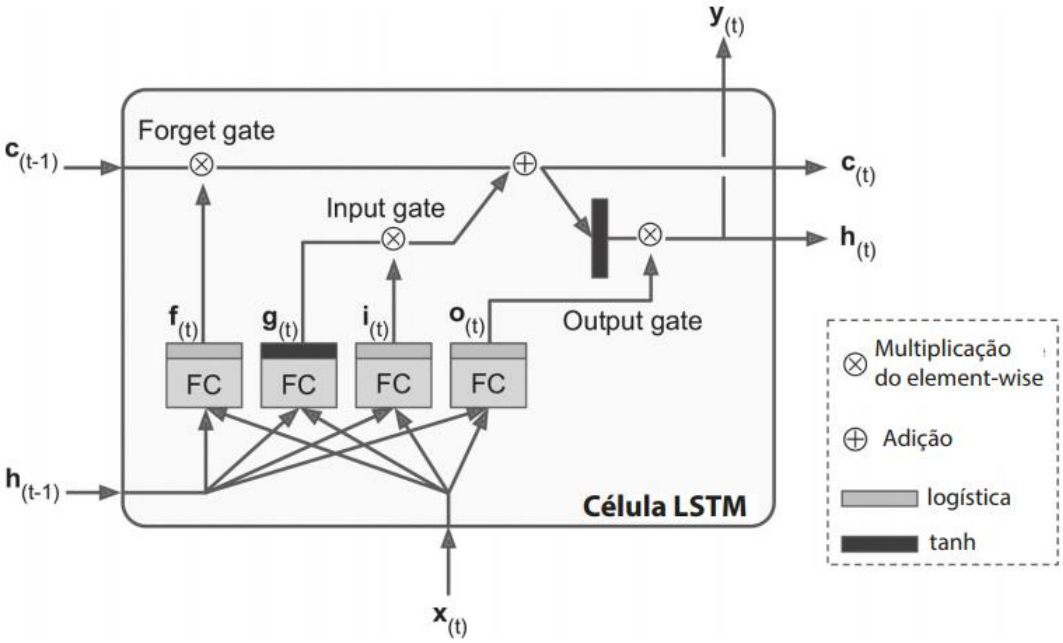
Fonte: [34]

2.5.2 Uma Célula LSTM

A LSTM é uma variação das redes neurais recorrentes (RNN), com uma Longa memória de curto prazo (LSTM), é eficiente na predição de séries temporais longas, essa arquitetura se difere de outras por possuir em laço interno de realimentação *(*feedback*)* e uma célula de memória, que retem informações do passado por mais tempo.

A Célula LSTM Foi proposta foi proposta em 1997 por Sepp Hochreiter e Jurgen Schmidhuber e aperfeiçoada em 2014 por Alex Graves, Haşim Sak e em 2015 por Wojciech Zaremba e sua estrutura básica está descrita na Figura 32, onde tem-se um comportamento básico, similar ao neurônio Perceptron da figura anterior, com uma modificação que é comparada a uma memória onde o estado da própria célula em uma iteração anterior representada pelo índice $_{(t-1)}$ influencia em um estado atual de índice $_{(t)}$ onde os sinais h_t funcionam como um estado de curto prazo e $c_{(t)}$ como um estado de longo prazo

Figura 32 - Célula LSTM



Fonte: [34]

A ideia-chave da utilização de células LSTM é que ao arranjá-las em rede, a mesma possa aprender o que armazenar no estado de longo prazo. Como o estado de longo prazo passa primeiro por um *forget gate*, a memória pode ser apagada ou renovada a cada iteração. Ao passo que um ajuste dessa memória pode ser feito através do *input gate* através da operação de adição. Já, a memória de curto prazo é a mesma saída da própria célula onde a memória de longo prazo é filtrada pela função tanh e multiplicada pela *output-gate*.

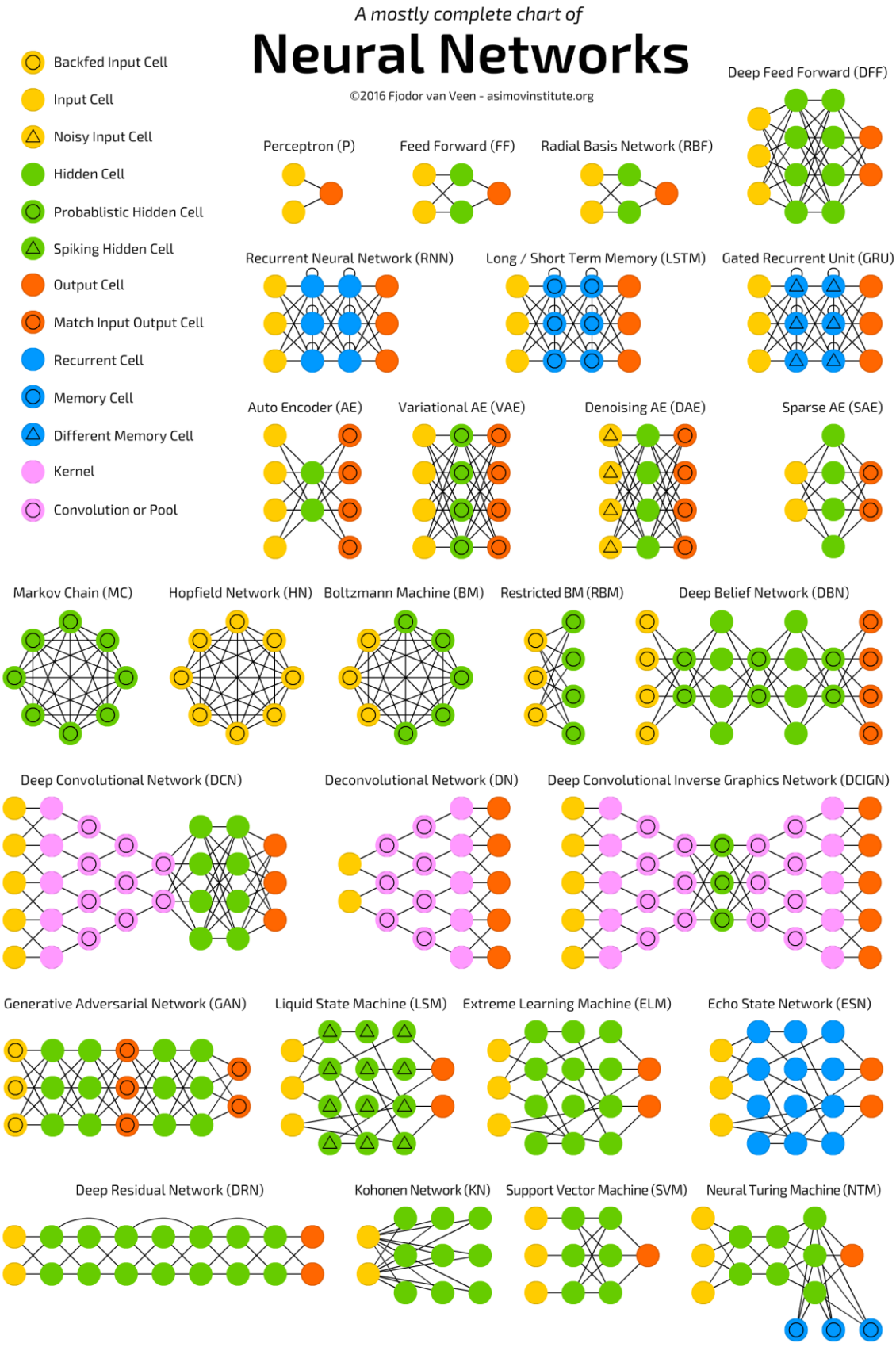
Uma célula LSTM pode aprender a reconhecer uma entrada importante, armazená-la no estado de longo prazo, aprender a preservá-la pelo tempo necessário e aprender a extraí-la sempre que for preciso. [34]

2.5.3 Uma Rede Neural

Ao agrupar-se estes neurônios ou células e interconectá-los em níveis chamados camadas obtém-se uma rede neural, ao variar: o formato do arranjo, o número de camadas, o tipo construção do neurônio, obtém-se arquiteturas diferentes que são especializados para resolver diferentes tipos de problemas.

As principais redes publicadas estão ilustradas na Figura 33.

Figura 33 - Redes Neurais



Fonte: [35]

2.5.4 Estrutura utilizada

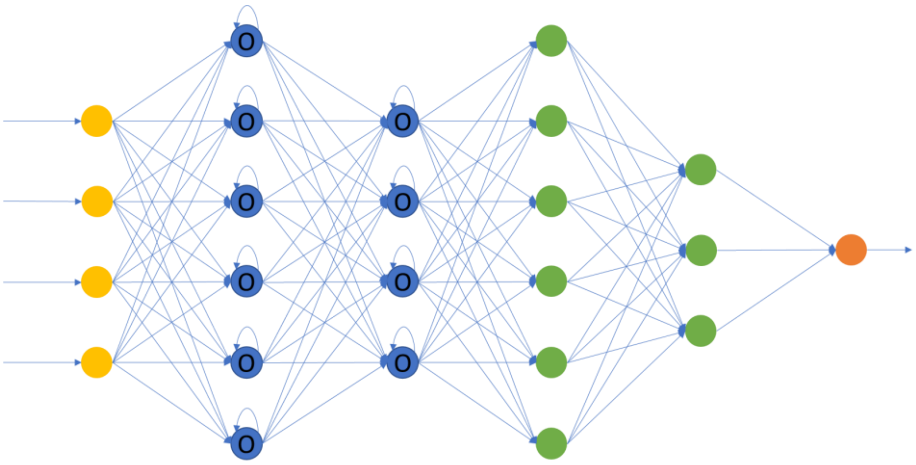
Neste estudo são avaliadas duas estruturas, primeiramente uma estrutura progressiva (FF, MLP) formada por múltiplas camadas de neurônios perceptron baseadas em [33], onde as previsões independem da sequência, dependendo apenas das variáveis de entrada, onde através destas se espera comprovar a dependência das características clássicas utilizadas para a previsão e condições esperadas naquele instante. Em seguida o estudo é refinado utilizando uma rede recorrente (RNN) com múltiplas camadas formadas por células LSTM associada a rede MLP. Onde a previsão anterior influi na imediata e a imediata influi na seguinte através de realimentações na rede.

No conjunto de camadas que forma a RN, a primeira camada é denominada camada de entrada e não processa nenhum cálculo, apenas serve de entrada para os parâmetros avaliados possuindo a mesma dimensão dos parâmetros de entrada, enquanto a última camada é a chamada camada de saída e entrega o valor final, neste caso, formada por apenas um neurônio.

Todas as camadas intermediárias entre as camadas de entrada e saída são chamadas camadas ocultas.

A Figura 34 ilustra um exemplo desta arquitetura de rede.

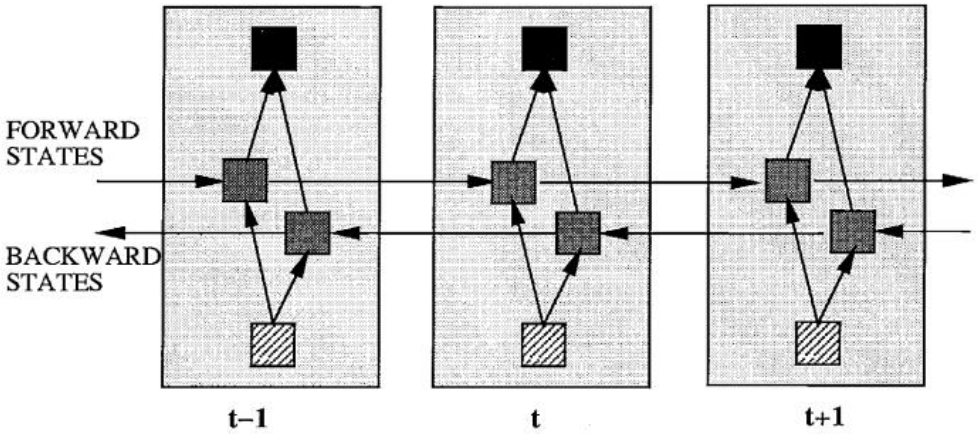
Figura 34 - Arquitetura LSTM+MLP



Fonte: Autor

Nesse estudo também será avaliada uma variação de RNN, chamada Bidirecional RNN (BRNN), essa estrutura proposta por Mike Schuster e Kuldip K. Paliwal [36], onde a célula utiliza tanto dados de um estado futuro $(t + 1)$, quanto de um estado passado $(t - 1)$ para determinar o estado (t).

Figura 35 - Rede BRNN



Fonte: [36]

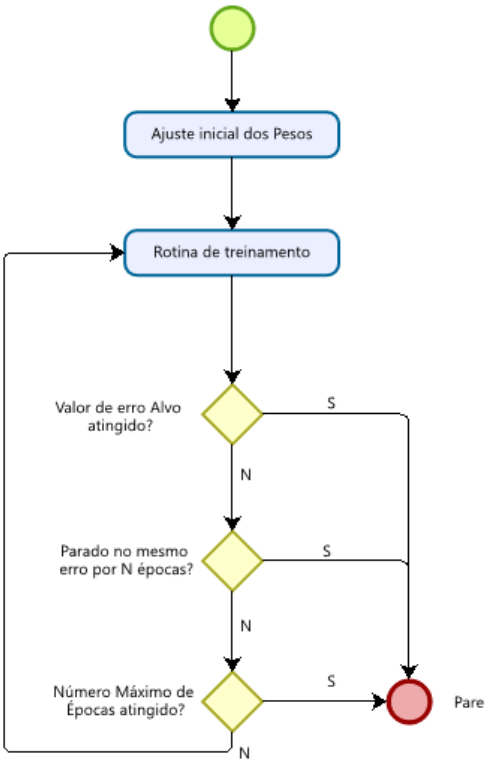
2.5.5 Aprendizagem

É denominado aprendizagem o processo de ajuste iterativo dos pesos que é realizado comparando os valores de saída conhecidos com os valores calculados, preditos, pela RNA, onde eles são atualizados por um processo de retro propagação do erro que é calculado e atualizado para cada ramo a cada interação, para este ajuste pode-se utilizar vários métodos. Este estudo encontrou melhores resultados utilizando o método Adam [37].

2.5.6 Critério de Parada

O processo de aprendizagem é repetido até que seja atingido um critério de parada que pode ser tanto por um número máximo de interações (épocas) ou pelo atingimento de um valor de precisão ou erro alvo pré-determinado ou ainda se após um número predefinido de iterações o erro não diminuir (Figura 36), este critério de parada também pode ser definido para evitar um sobreajuste do inglês *overfitting* (Sobre-ajuste ou sobreajuste é um termo usado em estatística para descrever quando um modelo estatístico se ajusta muito bem ao conjunto de dados anteriormente observado).

Figura 36 - Fluxograma de Parada



Fonte: Autor

2.5.7 Métrica de desempenho

Para avaliar o desempenho do modelo e permitir a comparação com outros trabalhos, foi utilizado o erro absoluto médio percentual (MAPE, *mean absolute percentage error)*, conforme a expressão (1).

$$\text{MAPE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} \frac{|y_i - \hat{y}_i|}{max(\epsilon, |y_i|)} \quad (1)$$

[1]

2.5.8 Implementação

Para implementação da MLP foi utilizado principalmente o módulo sklearn.neural_network.MLPRegressor da biblioteca *Scikit Learn.*

Com os dados normalizados pelo módulo *sklearn.preprocessingStandardScaler* da biblioteca *Scikit Learn.* Essa estrutura foi testada extensivamente utilizando uma estrutura de Pipeline.

Já, a rede LSTM foi implementada através do módulo *Keras* da biblioteca *TensorFlow*.

2.5.9 Ajuste de hiper parâmetros

O Ajuste dos hiperparâmetros foram realizados em três metodologias e três momentos distintos:

Momento 1: Para uma primeira abordagem utilizou-se a biblioteca modulo *sklearn.model_selection.GridSearchCV* da biblioteca *Scikit Learn*, a qual faz uma busca de todas as combinações possíveis de todos os parâmetros determinados. O que logo foi abandonado, uma vez que levaria-se muito tempo para percorrer todos os ajustes com o hardware disponível.

Momento 2: Para ter maior controle de cada ajuste foi elaborada uma função similar a estratégia do *GridSearchCV,* entretanto em uma função de laço que permitiu a análise de cada parâmetro de uma maneira semiautomatizada.

Momento 3: Como para a definição das dimensões da rede não se encontrou uma referência determinística. Optou-se por uma estratégia de otimização por algoritmo genético que é descrita na seção2.6.

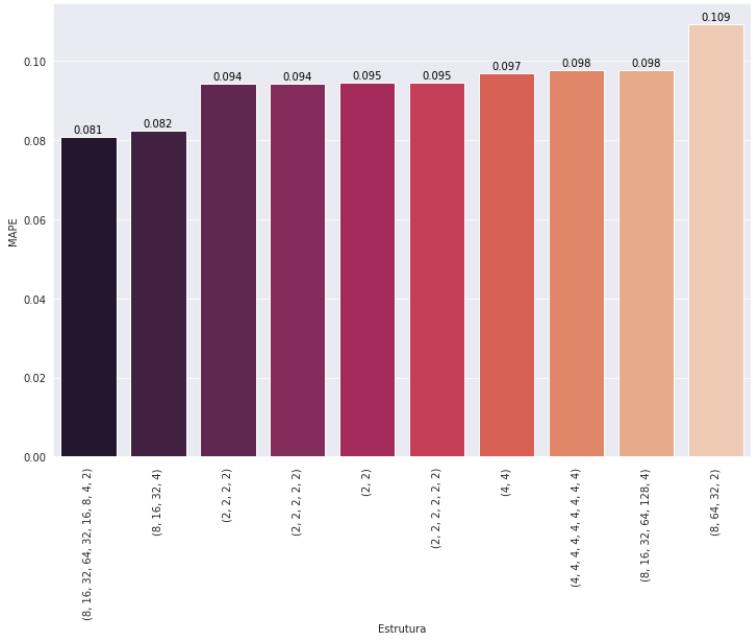
Camadas Ocultas *(hidden_layer_size)*

Foram pesquisadas 128 estruturas diferentes com profundidade máxima de 10 camadas ocultas e com o número de neurônios variando em potências de base 2 (2^n), com o número máximo de 1024 neurônios por camada. Exemplos de estruturas testadas:

- Retangular: (8, 8, 8, 8, 8, 8, 8, 8, 8, 8);
- Triangular Crescente: (8, 16, 32, 64, 128, 256, 4);
  - Triagular Decrescente: (128, 64, 32, 16, 8, 4, 2) e
- Diamante: (8, 16, 32, 64, 128, 32, 16, 8, 4, 2).

De todas as estruturas testadas as 10 que obtiveram melhor desempenho estão representadas na Figura 37.

Figura 37 - Desempenho por Estrutura



Fonte: Autor

Por ter um desempenho muito próximo e por possuir uma profundidade muito menor que o primeiro (4 vs 9), se optou pela estrutura (8,16,32,4).

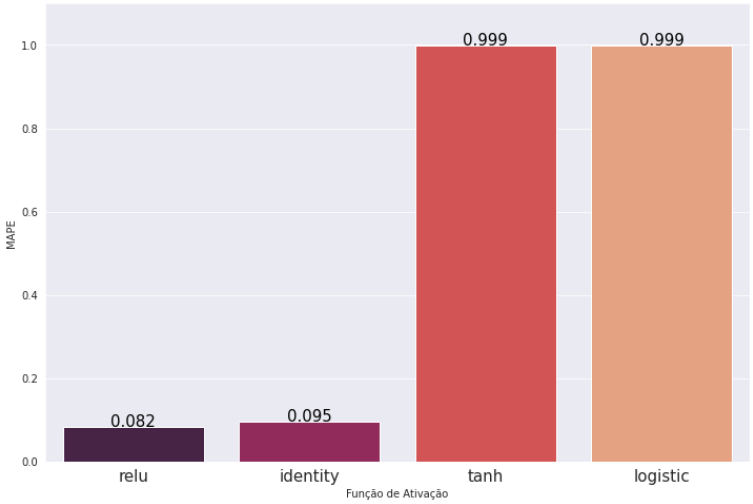
Função de ativação da camada oculta(*activation*)

Foram pesquisadas as 4 funções de ativação disponíveis:

- *identity*, $f(x) = x$
- *logistic*, $f(x) = \frac{1}{1 + exp(-x)}$
- *tanh*, $f(x) = tanh(x)$
- *relu*, $f(x) = max(0, x)$

De todas as funções de ativação testadas a que teve melhor desempenho foi a *relu* conforme representadas na Figura 38

Figura 38 - Desempenho por Função de Ativação



Fonte: Autor

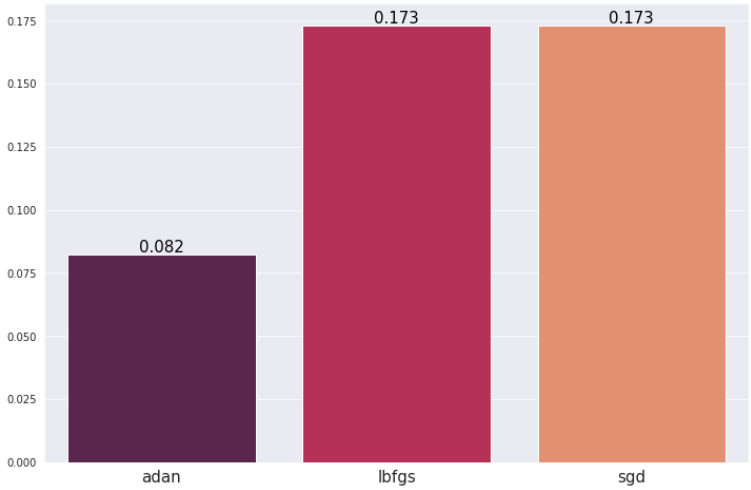
Otimizador de pesos (*Solver)*

Foram pesquisadas os 3 otimizadores disponíveis:

- *lbfgs, quasi-Newton methods.*
- *sgd*, Gradiente descendente
- *adam*, um método estocástico *gradient-based* proposto por Kingma, Diederik, e Jimmy Ba [37]

Dos 3 métodos avaliados o *adam*, obteve o melhor desempenho conforme Figura 39.

Figura 39 - Desempenho por otimizador



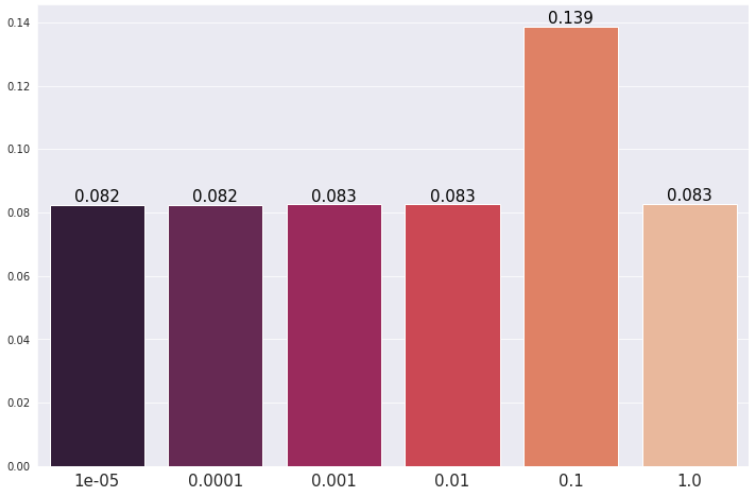
Fonte: Autor

Alfa (*alpha*)

Foram pesquisados valores de alfa: (1,0.1,0.01,0.001,0.0001,0.00001)

Como a alteração não trouxe melhor de desempenho (Figura 40), foi mantido o valor *default* = 0,0001

Figura 40 - Desempenho por alpha



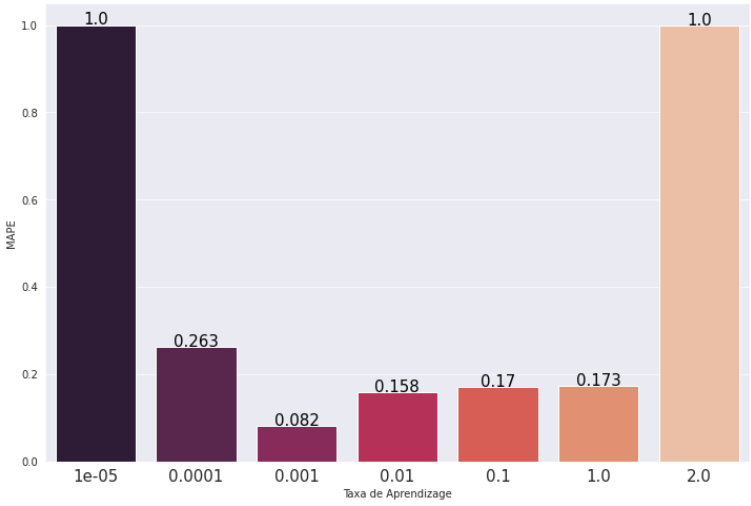
Fonte: Autor

Taxa de Aprendizagem (*learning_rate_ini*)

Foram pesquisados valores de taxa de aprendizagem: (0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0, 2.0)

Como a alteração não trouxe melhor de desempenho (Figura 41), foi mantido o valor *default* = 0,001

Figura 41 - Desempenho por Taxa de Aprendizagem



Fonte: Autor

Número Máximo de interações, épocas (*max_iter)*

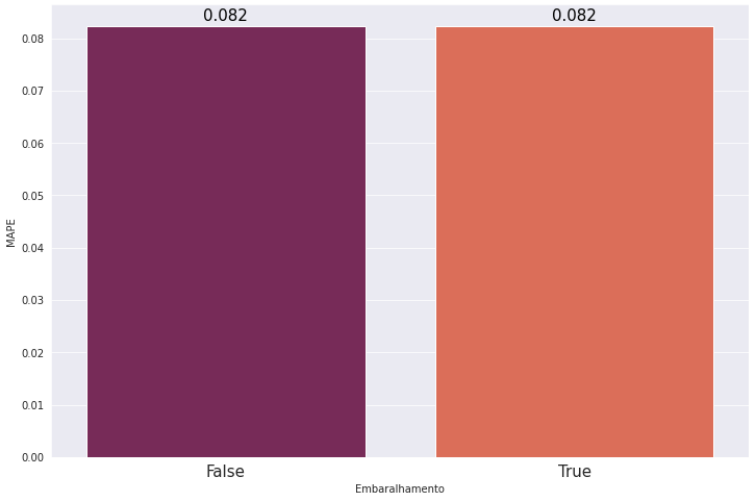
Número máximo de iterações utilizadas = 100.000

Embaralhamento (*shuffle*)

Foram pesquisados de embaralhamento dos dados de entrada: {*True, False*}

Como a alteração não trouxe melhoria de desempenho (Figura 42), foi mantido o valor *default = True*

Figura 42 - Desempenho por embaralhamento



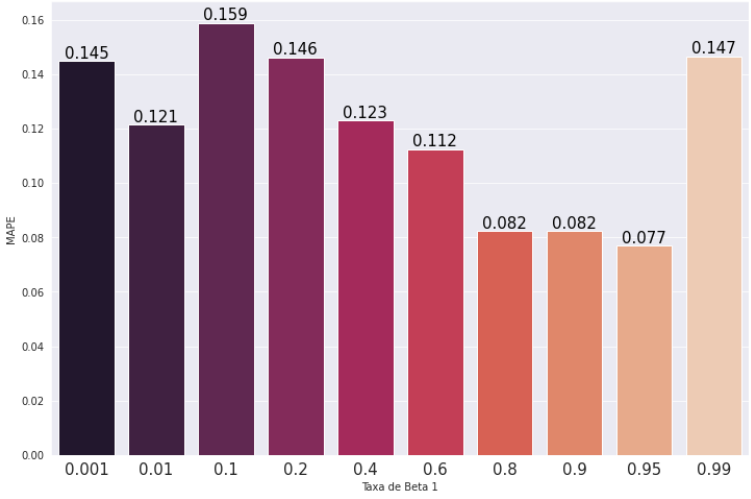
Fonte: Autor

Beta 1

Parâmetro 1 para ajuste de Bias

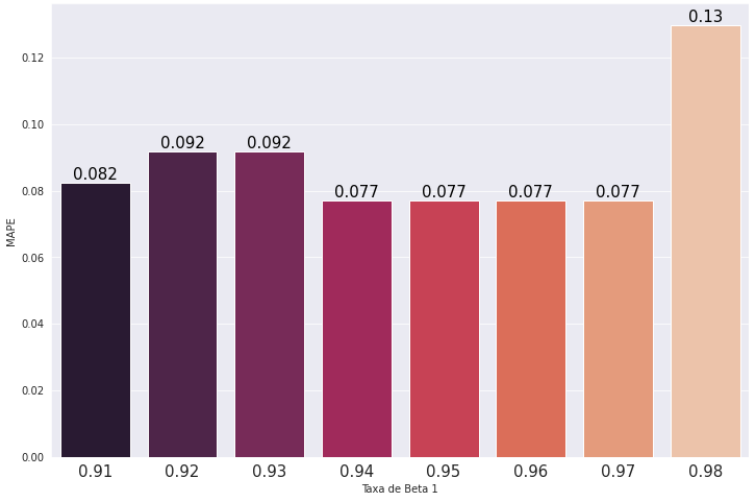
Foram pesquisados valores de Beta 1: (0.001, 0.01, 0.1, 0.2, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99) e em seguida de (0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98. O melhor resultado foi em Beta 1 = 0.94 conforme Figura 43 e Figura 44.

Figura 43 - Desempenho por Beta 1



Fonte: Autor

Figura 44 - Desempenho por Beta 1(Ajuste 2)



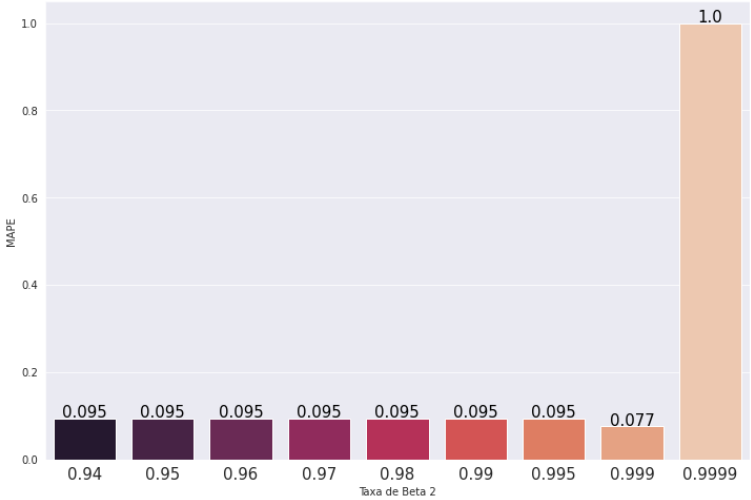
Fonte: Autor

Beta 2

Parâmetro 2 para ajuste de Bias

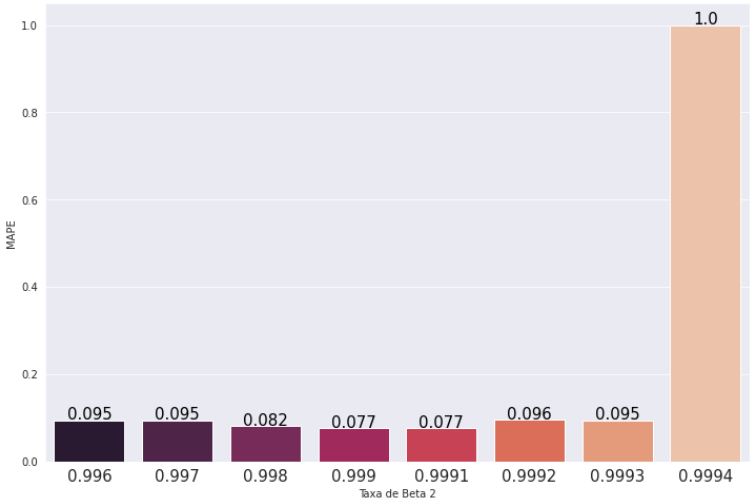
Foram pesquisados valores de Beta 2: (0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 0.995, 0.999, 0.9999), em seguida buscou-se um refinamento avaliando os parâmetros: (0.996, 0.997, 0.998, 0.999, 0.9991, 0.9992, 0.9993, 0.9994). O melhor resultado foi encontrado no valor *default* = 0.999, o qual foi o escolhido mantido conforme Figura 45 e Figura 46.

Figura 45 - Desempenho por Beta 2



Fonte: Autor

Figura 46 - Desempenho por Beta 2(Ajuste 2)



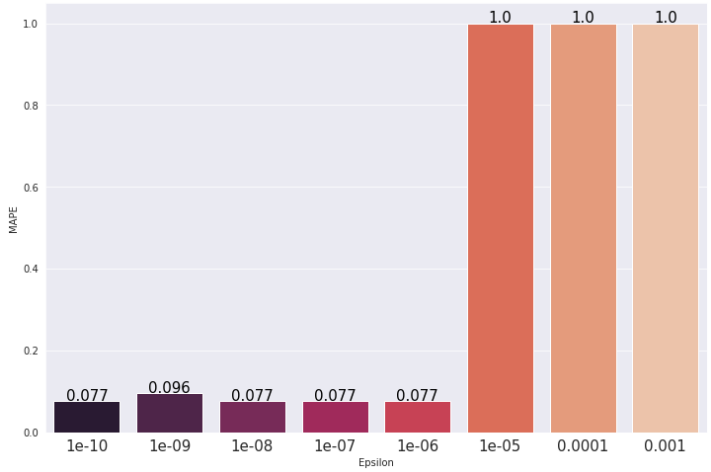
Fonte: Autor

Epsilon

Valor numérico de estabilidade no algoritmo *adam*

Foram pesquisados valores de estabilidade *epsilon*: (1-e10, 1-e09, 1-e08, 1-e07, 1-e06, 1-e05, 1-e04, 1-e03). O melhor resultado foi em encontrado no valor 1-e10 o qual foi o escolhido mantido conforme Figura 47.

Figura 47 - Desempenho por Epsilon

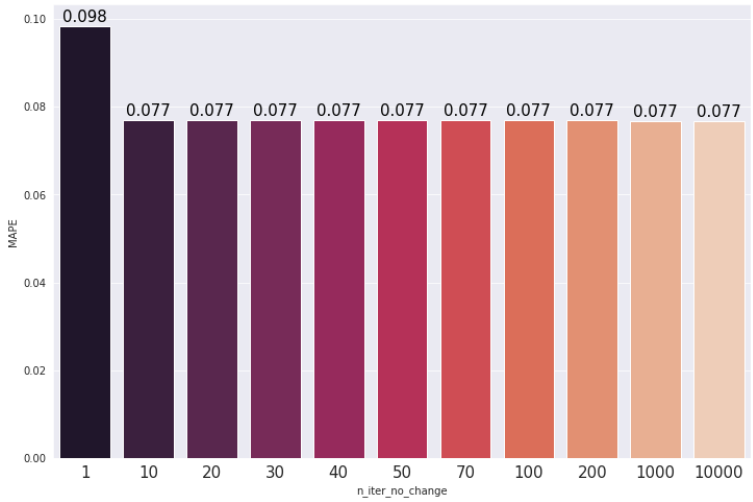


Fonte: Autor

Número de épocas máximas sem mudança (*n_iter_no_change*)

Foram pesquisados valores de número de épocas mínimas para critério de parada sem mudanças (1,10,20,30,40,50,70,100,200,1000,10000,100000) e obteve-se o melhor resultado em 1000 conforme Figura 48.

Figura 48 - Desempenho por número de interações sem mudança



Fonte: Autor

2.6 Otimização por Algoritmo genético

O Algoritmo genético é uma técnica de otimização, baseada nos princípios de seleção natural, onde a partir de uma população inicial e uma avaliação de aptidão os indivíduos mais aptos são selecionados e geram descendentes, propagando seus genes. Os menos aptos morrem sem deixar descendentes e os descendentes podem ou não sofrer algum tipo de mutação aleatorizada. Neste caso cada indivíduo representa uma arquitetura para a Rede Neural e alguns hiperparâmetros, e a aptidão é a medida do menor MAPE

- **Seleção**

Seleção é o processo pelo qual uma certa proporção de indivíduos é selecionada para acasalar entre si e gerar novos descendentes. Assim como na seleção natural da vida real, os indivíduos mais aptos têm maiores chances de sobreviver e, portanto, de passar seus genes para a próxima geração. Embora existam versões com mais indivíduos, geralmente o processo de seleção combina dois indivíduos, criando pares de indivíduos. Onde existem quatro estratégias principais, descritas a seguir.

1. Emparelhamento: Consiste em emparelhar os cromossomos mais aptos, dois a dois, e realizar o cruzamento de genes seguindo uma regra preestabelecida ou de maneira aleatória.

2. Aleatório: Esta estratégia consiste em selecionar indivíduos aleatoriamente do pool de acasalamento.

3. Roleta: Esta estratégia também segue um princípio aleatório, mas os indivíduos mais aptos têm maiores probabilidades de serem selecionados.

4. Torneio: com essa estratégia, o algoritmo primeiro seleciona alguns indivíduos como candidatos e, em seguida, seleciona o indivíduo mais apto. Esta opção tem a vantagem de não exigir que os indivíduos sejam classificados primeiro por aptidão.

Para este experimento é utilizada a estratégia "roleta".

- **Delimitação do problema**

Como na busca anterior, foi encontrada como melhor arranjo a configuração de camadas ocultas a configuração (8,16,32,4) e considerando que para cada avaliação de aptidão será necessário treinar e validar a RNA. Definiu-se como espaço de busca uma rede de até 4 camadas ocultas com dimensão de no máximo 64 neurônios por camada. Como a quantidade de neurônios é quantizada, cada camada poderá ter apenas números inteiros de N neurônios. Ou seja, cada indivíduo terá um cromossomo representado por um conjunto de quatro genes: {N,N,N,N} onde $N = \{0,1,2, \dots ,64\}$

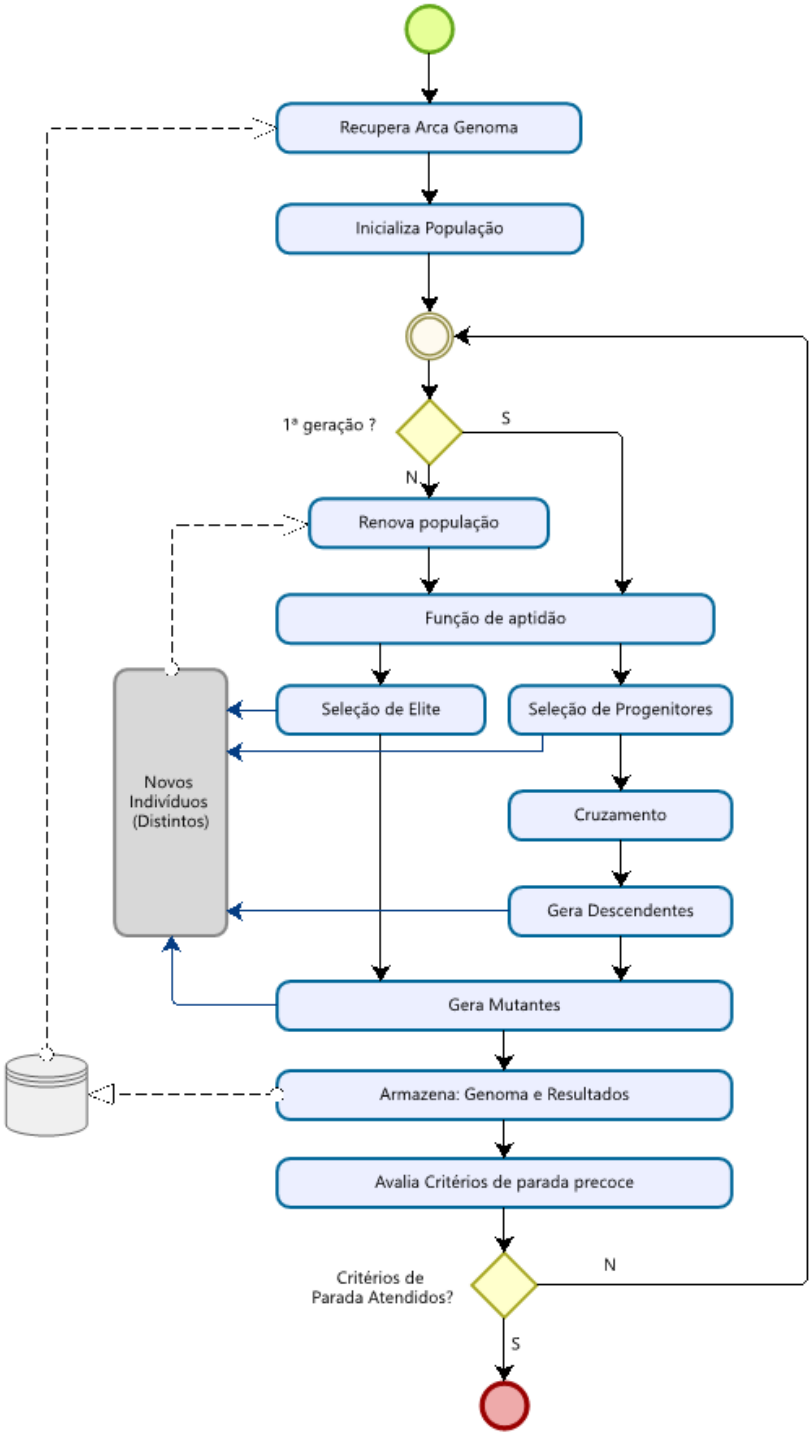
Onde um treinamento levou em média 3 minutos. Já o conjunto de todas as combinações possíveis é dado por: $65^4 = 17.850.625\ combinações$.

O custo em tempo de calcular todas as possibilidades é estimado em 53.551.875 minutos ou 892.531 horas ou 37.188,8 dias ou 101.817 anos. O que por si já é a justificativa de um método de otimização que encontre possíveis atalhos.

O método implementado é inspirado em [38], e descrito pelo fluxograma da Figura 49.

*obs.: Aleatoriamente - Refere-se a número pseudo aleatório gerado pela linguagem de programação

Figura 49 - Fluxograma Algoritmo Genético



Fonte: Autor

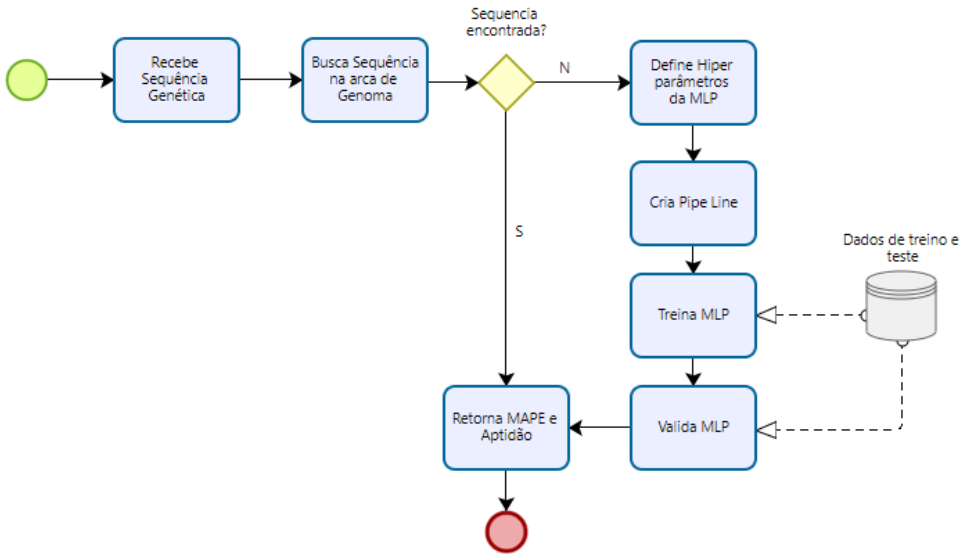
- **Arca de Genoma**

Para evitar que um mesmo código genético seja testado mais de uma vez, foi criada uma rotina que armazena em arquivo e recupera os valores de erro e aptidão para cada rodada completa do algoritmo.

- **Função de aptidão**

Para cada sequência gerada é realizado o teste de aptidão que parametriza e testa a MLP para um mesmo caso de teste, retornando os valores de aptidão. Caso a mesma sequência já tenha sido testada o algoritmo recupera o valor do teste anterior promovendo assim um ganho de performance (Figura 50).

Figura 50 - Fluxograma Função de Aptidão

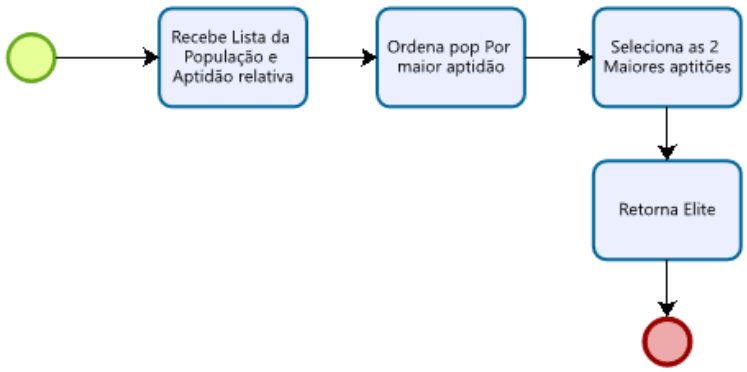


Fonte: Autor

- **Elite**

Para garantir que o mais apto seja mantido nos processos de mutação na próxima geração, são selecionados os dois indivíduos mais aptos garantindo sua sobrevivência e mantendo o menor erro no laço para avaliação do processo de parada (Figura 51).

Figura 51 - Fluxograma Elite

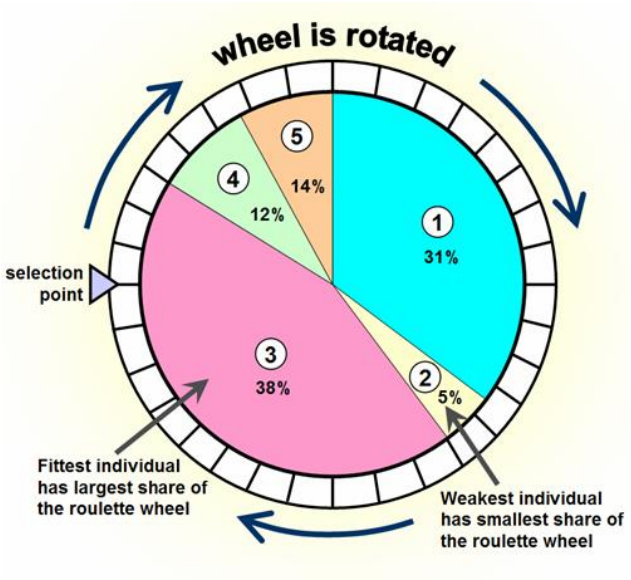


Fonte: Autor

- **Progenitores**

Após testados cada indivíduo possui uma aptidão medida, essa aptidão é normalizada pela soma de todas as aptidões da população atual, e esta aptidão normalizada é utilizada como distribuição de probabilidade para uma seleção aleatória, onde em uma analogia com uma roleta de jogos, os mais aptos tem a maior probabilidade de ser escolhido aleatoriamente* (Figura 52).

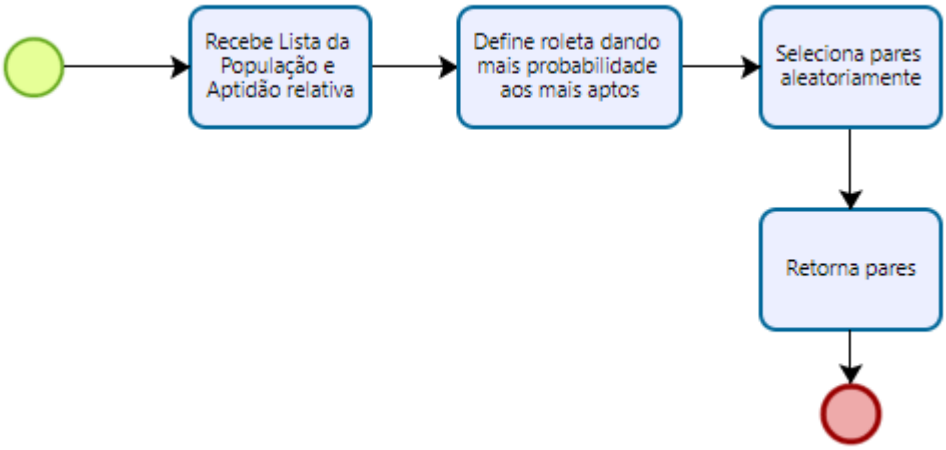
Figura 52 - Seleção de Progenitores por roleta ponderada



Fonte: [39]

Após selecionados os progenitores são agrupados aos pares para que possam realizar cruzamento genético (Figura 53)

Figura 53 - Fluxograma Seleção de progenitores

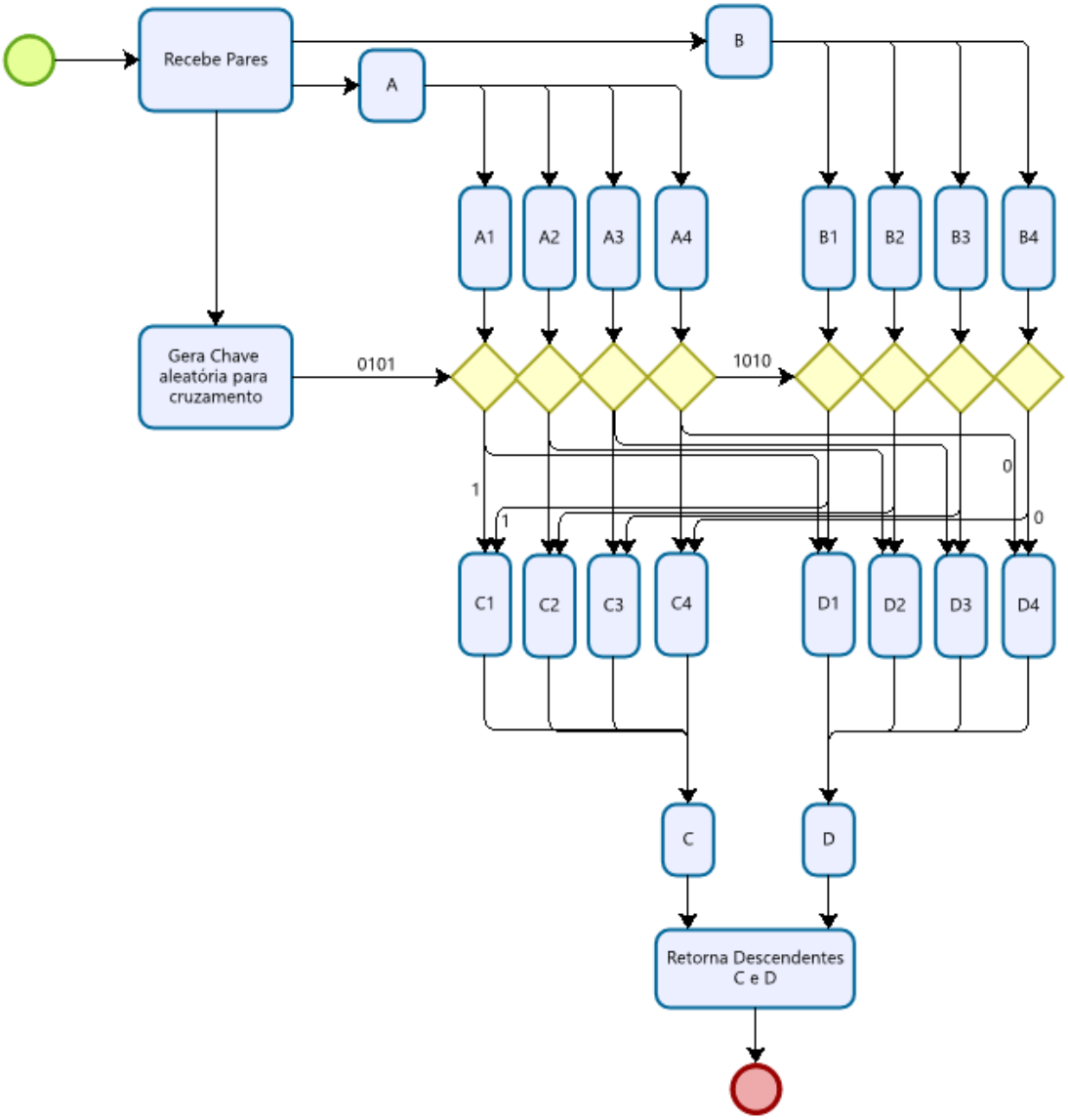


Fonte: Autor

- **Cruzamento**

Para no cruzamento evitar que sejam gerados clones, é gerada uma chave aleatória de 4 bits diferentes de “0000” e “1111” com a mesma probabilidade para cada posição ser 0 ou 1. Esta chave define qual descendente (C ou D) herda qual gene, de maneira que, nesta etapa, C é o inverso de D no que se refere a posição do gene herdado do progenitor A ou B(Figura 54).

Figura 54 - Fluxograma Cruzamento



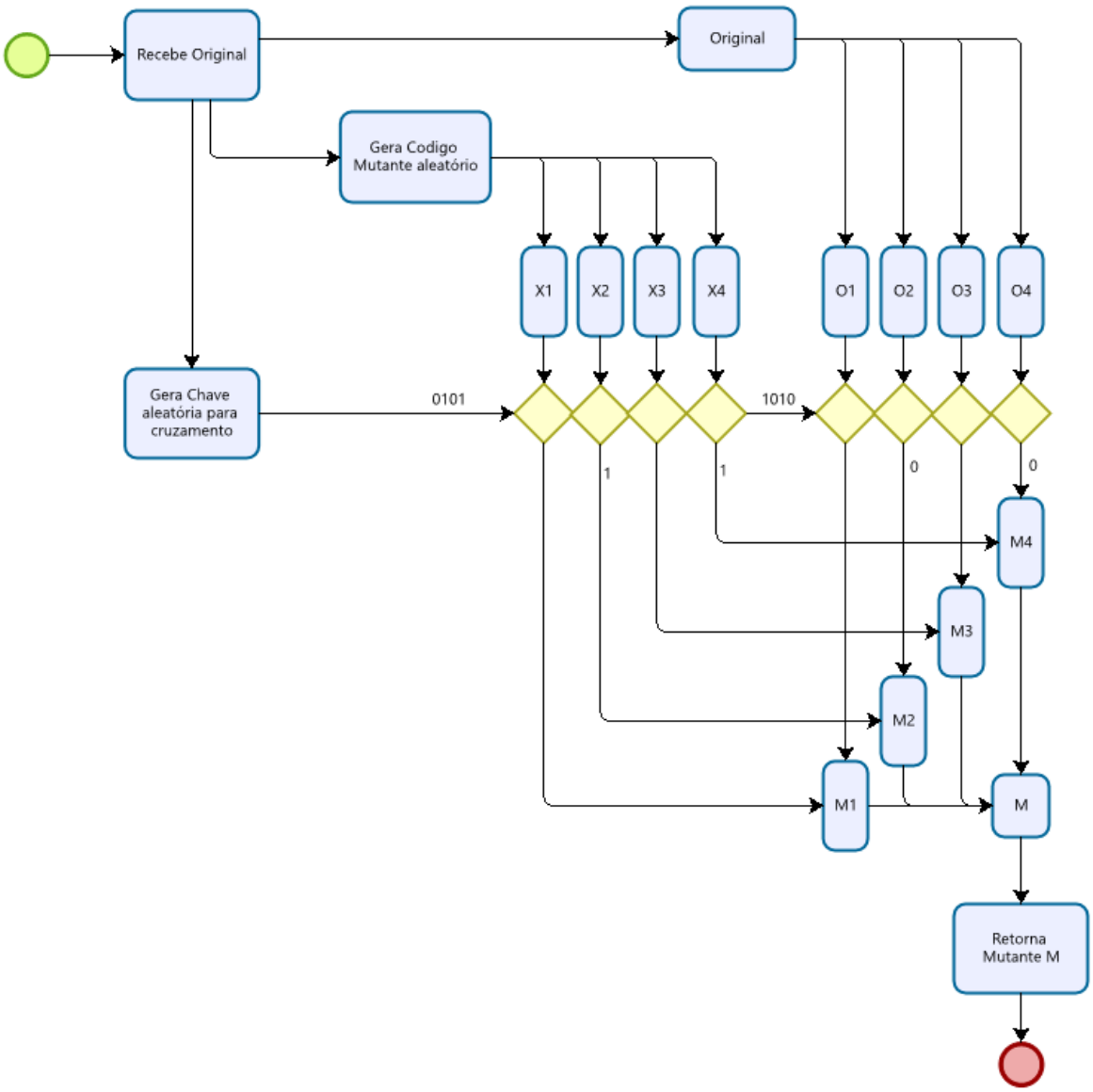
Fonte: Autor

- **Mutação**

Para garantir que novas sequencias sejam testadas a partir de um parâmetro de probabilidade de mutação inicial, é gerada uma chave sequencial de forma similar ao processo de cruzamento garantindo que haja ao menos uma mutação e um progenitor fictício gerado

aleatoriamente a cada mutação. Caso o algoritmo fique preso no mesmo erro mínimo, o fator de mutação é aumentado (Figura 55).

Figura 55 - Fluxograma Mutação



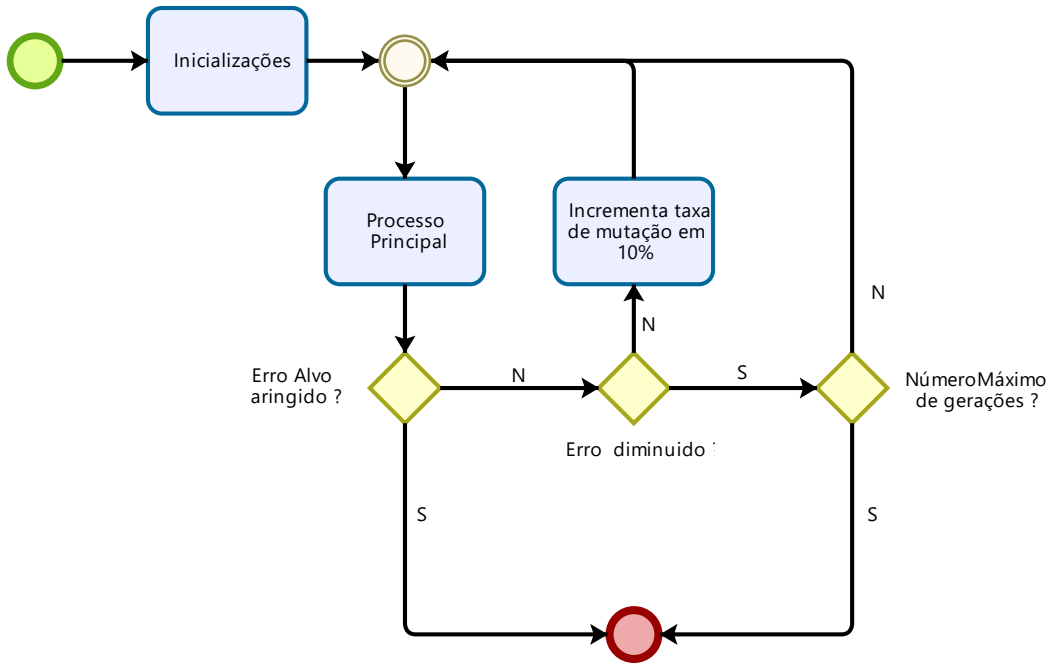
Fonte: Autor

- **Critérios de Parada**

  Como critérios de parada tem-se a seguinte sequência:

  - Parada por atingimento de alvo: caso o erro predefinido como alvo, seja atingido de maneira precoce;
  - Parada por erro travado: caso o erro fique parado por 4 gerações o fator de mutação é incrementado em 10% até o limite de 100%, se o erro persistir travado por um número de gerações maior que o predefinido então é realizada a parada.
  - Parada por número máximo de gerações: A parada ocorre após superado o número máximo de gerações predefinido.

Figura 56 - Fluxograma Critérios de Parada Algoritmo Genético

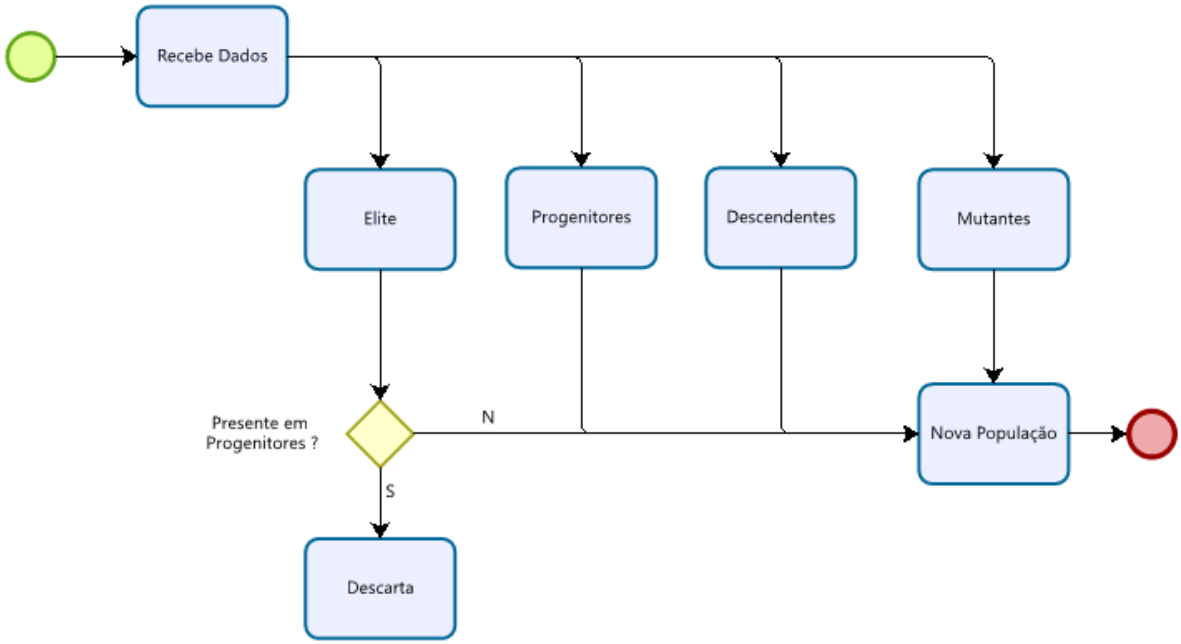


Fonte: Autor

- **Nova Geração**

A nova geração é formada pela composição dos progenitores, seus descendentes e versões mutadas aleatoriamente dos descendentes e da elite. Caso algum membro da elite não seja selecionado como progenitor o mesmo será incluído na próxima geração, para que seja garantida a métrica de ter o mais indivíduo com melhor aptidão sempre presente nas gerações seguintes (Figura 57).

Figura 57 - Fluxograma Nova Geração



Fonte: Autor

Como metade da população é selecionada como progenitora e cada par de progenitores gera um par de filhos que por sua vez cada descendente produzirá uma réplica mutante, e cada membro da elite (2 indivíduos), produzirá uma réplica mutante, teremos uma expansão da população da primeira para a segunda geração respeitando a seguinte regra:

$$iniPOP = População\ inicial \tag{2}$$

$$nova\ população = \frac{iniPOP}{2} + \frac{iniPOP}{2} + \frac{iniPOP}{2} + 2 \tag{3}$$

$$nova\ população = \frac{3}{2} iniPOP + 2 \tag{4}$$

Se a elite for pré-selecionada como progenitora

$$nova\ população = \frac{3}{2} iniPOP + 4 \tag{5}$$

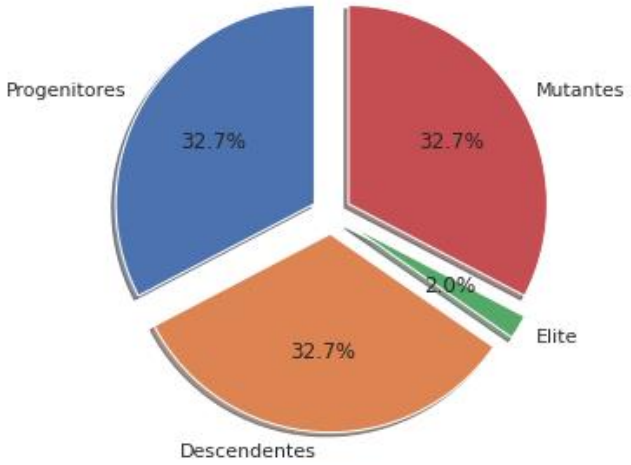
Se a elite não for pré-selecionada como progenitora

Assim, a população clímax oscila entre estes dois valores ou seja:

$$\frac{3}{2} iniPOP + 2 \leq População \leq \frac{3}{2} iniPOP + 4 \quad (6)$$

Conforme ilustrado na Figura 58

Figura 58 - Proporção da Nova Geração



Fonte: Autor

- **Resultados obtidos com algoritmo genético**

Cada rodada de teste poderia gerar menores dimensões de camadas ocultas, por exemplo uma sequência [3, 4, 0, 5] era parametrizada na pipeline como [3, 4, 5] uma vez que a MLP não permite camadas ocultas nulas, mas este artifício foi mantido para uma busca randômica de arquiteturas, entretanto para efeitos de comparação de performance, o algoritmo foi executado com as mesmas configurações para: 2, 3 e 4 camadas máximas. Este teste foi realizado para comparação mais justa uma vez que a probabilidade da camada ser anulada é de 1 em 9 para cada posição.

Parâmetros de comparação:

Número inicial de indivíduos = 64

Limites de valores de cada gene = [0, 64]

Número máximo de gerações = 100

Índice inicial de probabilidade de mutação = 20%

As melhores configurações para cada arquitetura estão mostradas na Tabela 3 -

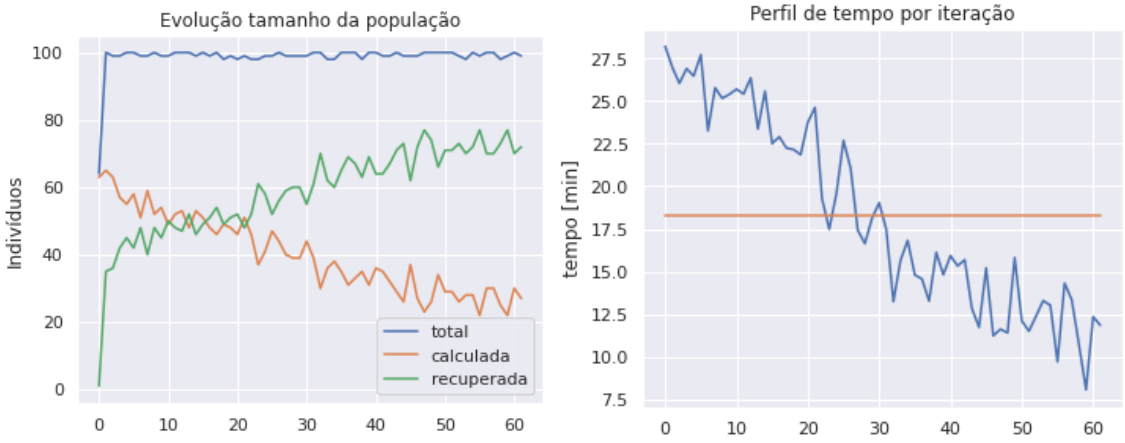
Tabela 3 - Resultados por números de camadas

Nº Máx. Camadas	*Menor Erro (MAPE)*	*Cromossomo*
2	6,707%	[4, 5]
3	5,769%	[5, 3, 35]
4	5,237%	[2, 3, 59, 38]

Fonte: Autor

Avaliação do ganho de performance do armazenamento dos resultados anteriores, pode ser melhor verificada no teste com até duas camadas uma vez que neste se testou, neste lote, 60 % das combinações possíveis onde na Figura 59 pode-se evidenciar uma redução do tempo de processamento de cada iteração (linha azul no gráfico da direita) e um aumento dos resultados recuprerados (gráfico da esquerda).

Figura 59 - Desempenho por iteração teste até 2 camadas

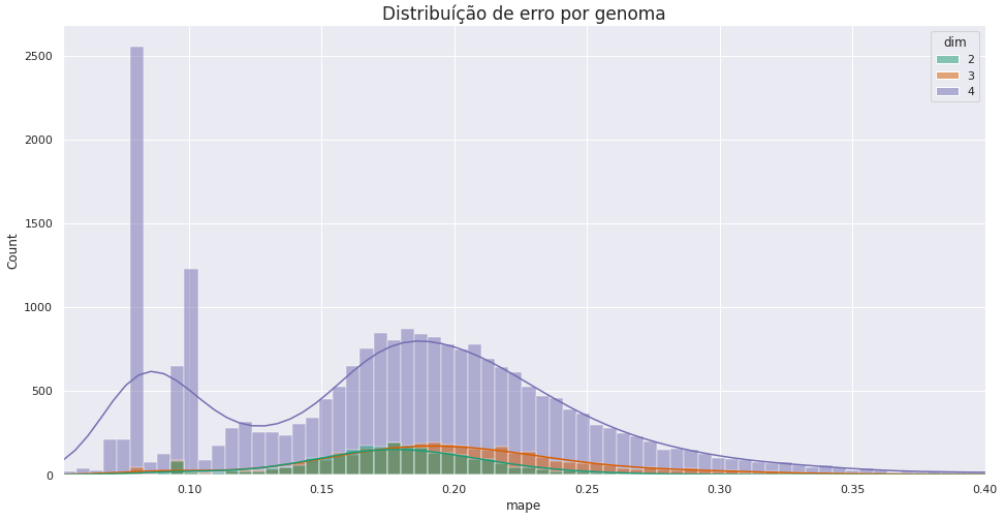


Fonte: Autor

Outro aspecto relevante a ser avaliado é a dispersão dos resultados encontrados para 30 mil combinações testadas, o que representa 0,17% das combinações possíveis, onde ao ajustar aleatoriamente a MLP o melhor desempenho obtido foi o de 5,23% . Lembrando que esse valor está tendendo ao menor valor uma vez que o algoritmo genético cria mais descendentes do

indivíduo mais apto o que fica evidenciado pala maior barra a esquerda (em torno de 7,79% e 8,29%).

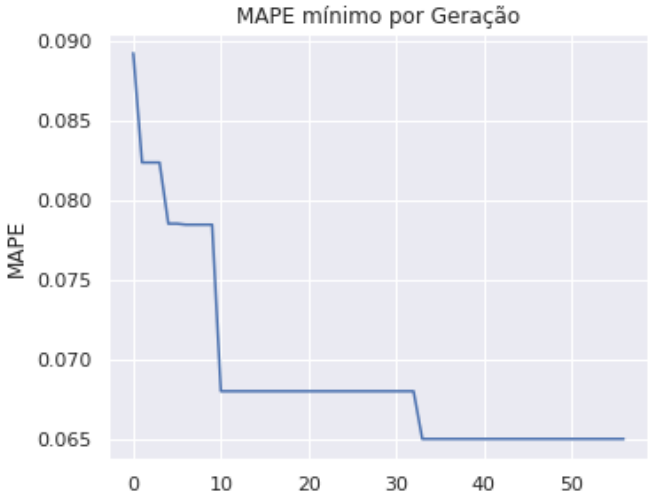
Figura 60 - Distribuição de erro por genoma



Fonte: Autor

Também forma realizados testes que iniciaram com pelo menos um indivíduo dentre o segundo e o décimo melhor testado, nesse caso o algoritmo conseguiu encontrar o melhor resultado em até 3 iterações, já ao iniciar randomicamente os testes mais lentos levaram em torno de 32 gerações para atingir um resultado muito próximo do melhor encontrado até o momento, considerando uma probabilidade de mutação fixa de 20%, alterando iterativamente este valor pode se obter o resultado com menos gerações.

Figura 61 - Evolução do erro por geração



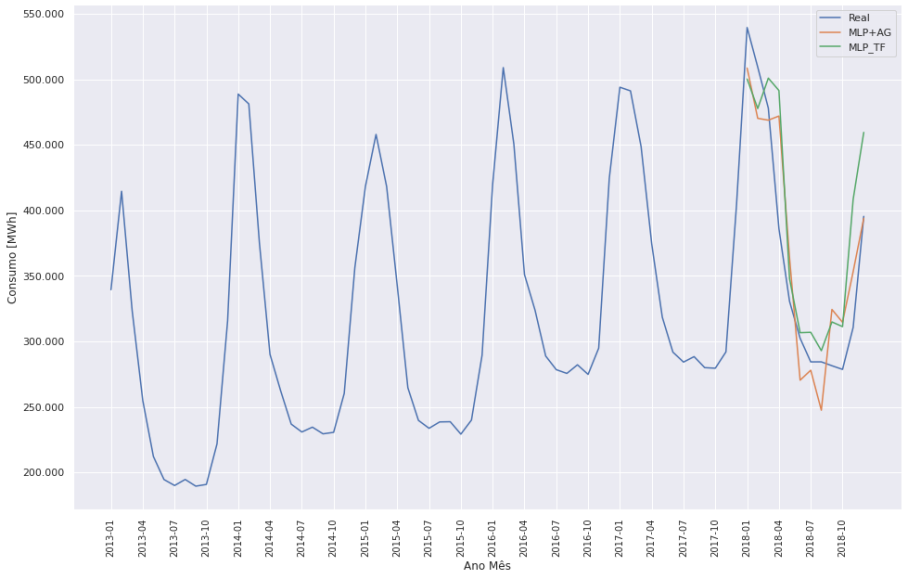
Fonte: Autor

2.7 Mudança de plataforma do *SckitLearn* para *TensorFlow*

Após estudo da MLP limitações e a exploração dos hiperparâmetros, verificou-se que a biblioteca *TensorFlow* seria mais adequada para o desenvolvimento, por possuir suporte a GPU (Unidade de Processamento Gráfico, do inglês: *Graphics Processing Unit*) e TPU, que são unidades de processamento que obtém um melhor desempenho que as CPUs (Unidade Central de Processamento, *do inglês: Central Process Unit)* no treinamento em técnicas de aprendizagem profunda, além de e a implementação da LSTM ser facilitada a partir do modulo *Keras*. Entretanto a estrutura pipeline teve de ser modificada o módulo de normalização alterada de *Standarscaler* para a *MinMaxScaler* (ambas pertencentes a *Scikitlearn*.

Em um teste preliminar com até 100.000 épocas e parada em 10 resultados iguais, o *TensorFlow* foi mais rápido convergindo em 1457, enquanto o *Scikitlearn* levou 9528, mas com uma precisão em torno de 1% melhor. Graficamente as respostas são similares, conforme pode-se verificar na Figura 62, ainda poder-se-ia fazer uma melhor ajuste, entretanto como o foco nessa etapa foi o de o de implementação da LSTM, maiores refinamentos são realizados, quando necessário, em etapas posteriores.

Figura 62 - Scikitlearn vs TensorFlow



Fonte: Autor

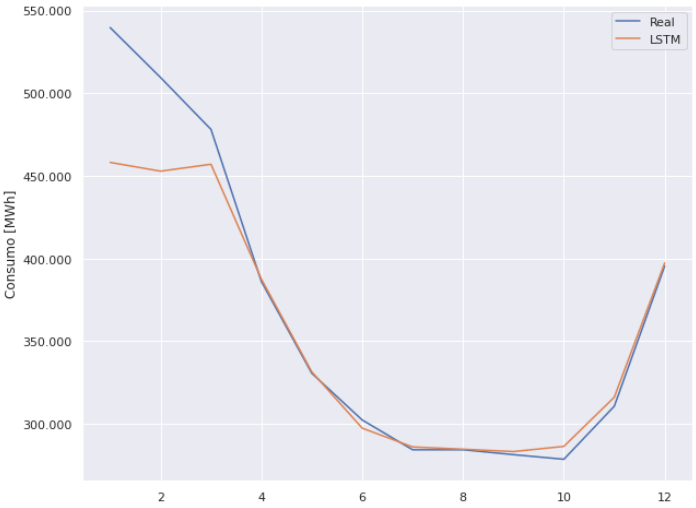
- **Implementação da estrutura LSTM**

Seguindo o desenvolvimento da rede LSTM, foi utilizada a biblioteca *Keras* e o algoritmo genético discutido anteriormente para definir o arranjo dessa estrutura, foram testadas variações de até 6 camadas ocultas de profundidade com 3 variações de *dropout* que é quando se desliga (aleatoriamente) algumas interconexões, portanto a as conexões entra as camadas

poderia ser de três formas: totalmente interconectada; 90% interconectada (*dropout* = 0,1); ou 80% interconectada (*dropout* = 0,2).

Cada uma das camadas poderia ter até 64 neurônios (onde 0 (zero) descartaria a camada), onde as duas primeiras poderiam assumir duas formas LSTM (unidirecional) ou BLSTM (bidirecional LSTM), as quatro últimas camadas poderiam assumir apenas o formato MLP, após testes descritos em Anexo 4 – Resultado Teste: 2 camadas (Bi) LSTM e 4 camadas MLP com *drop*out, onde foi possível aproximar a série de 2018 com um erro (MAPE) de 0,0327 o que é um resultado melhor que o obtido apenas com a estrutura anterior apenas com camadas MLP (Figura 63).

Figura 63 - Teste Valor Real vs Predito LSTM



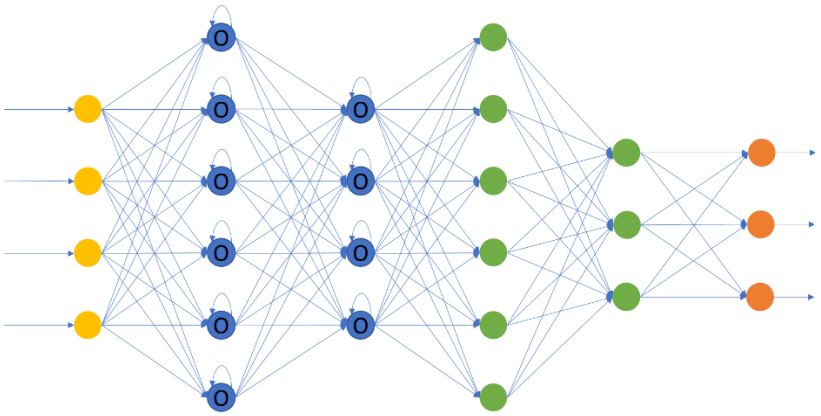
Fonte: Autor

- **Generalização LSTM**

Como este estudo foi segmentado inicialmente para viabilizar uma análise mais profunda separando para isso somente o consumidor rural no estado do Rio Grande do Sul. São realizadas duas generalizações. A primeira generalizando para qualquer classe de consumidor, e a segunda para qualquer estado e brasil;

Analisando as classes de consumo foram realizados dois testes o primeiro prevendo todas as classes de consumo e o consumo total em uma única rede com nove saídas. Esta rede foi submetida ao mesmo procedimento de ajuste via algoritmo genético o qual está melhor documentado nos Anexo 5 – Resultado Teste: 2 camadas (Bi) LSTM e 4 camadas MLP com *Dropout* caso: RS Estratificado Com Setores N1, Rede Única para Todos os Setores. Figura 64 de forma que uma serie de consumo pudesse interferir diretamente no outro.

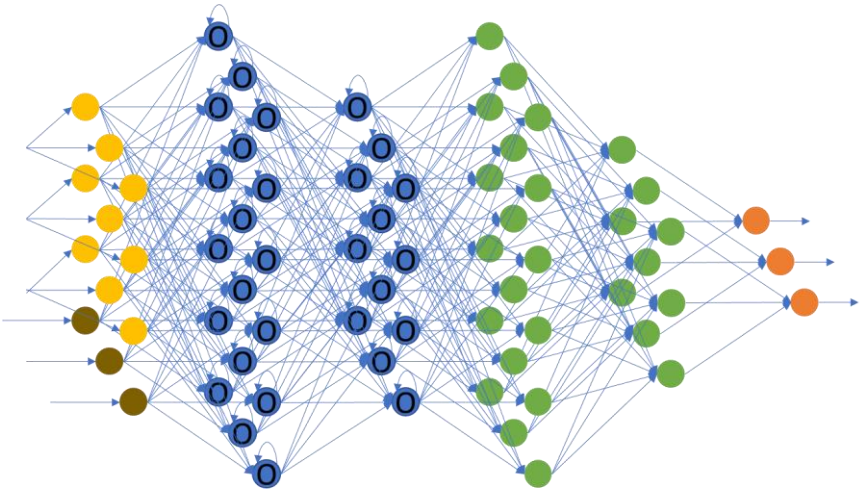
Figura 64 - Arquitetura Interligada LSTM+MLP



Fonte: Autor

Em seguida foi realizado o teste de ajuste cada SetorN1 individualmente, ou seja, uma rede com apenas uma saída em uma estrutura paralela onde foram compartilhados apenas as outras entradas (que não são as classes de consumo) conforme descritos na Figura 65. Ambos os processos foram ajustados para que demorassem em torno de 24h.

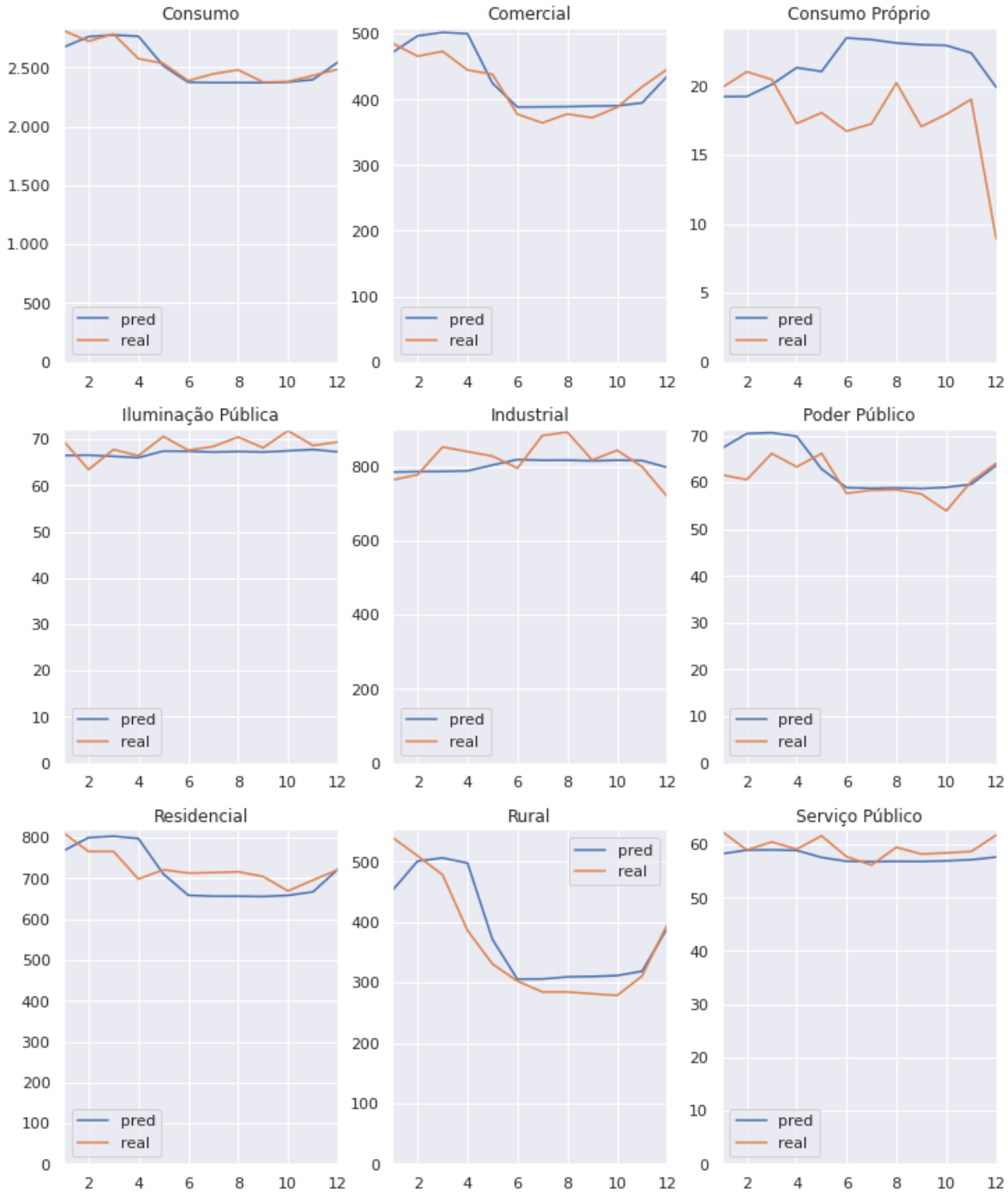
Figura 65 - Arquitetura Paralela LSTM+MLP



Fonte: Autor

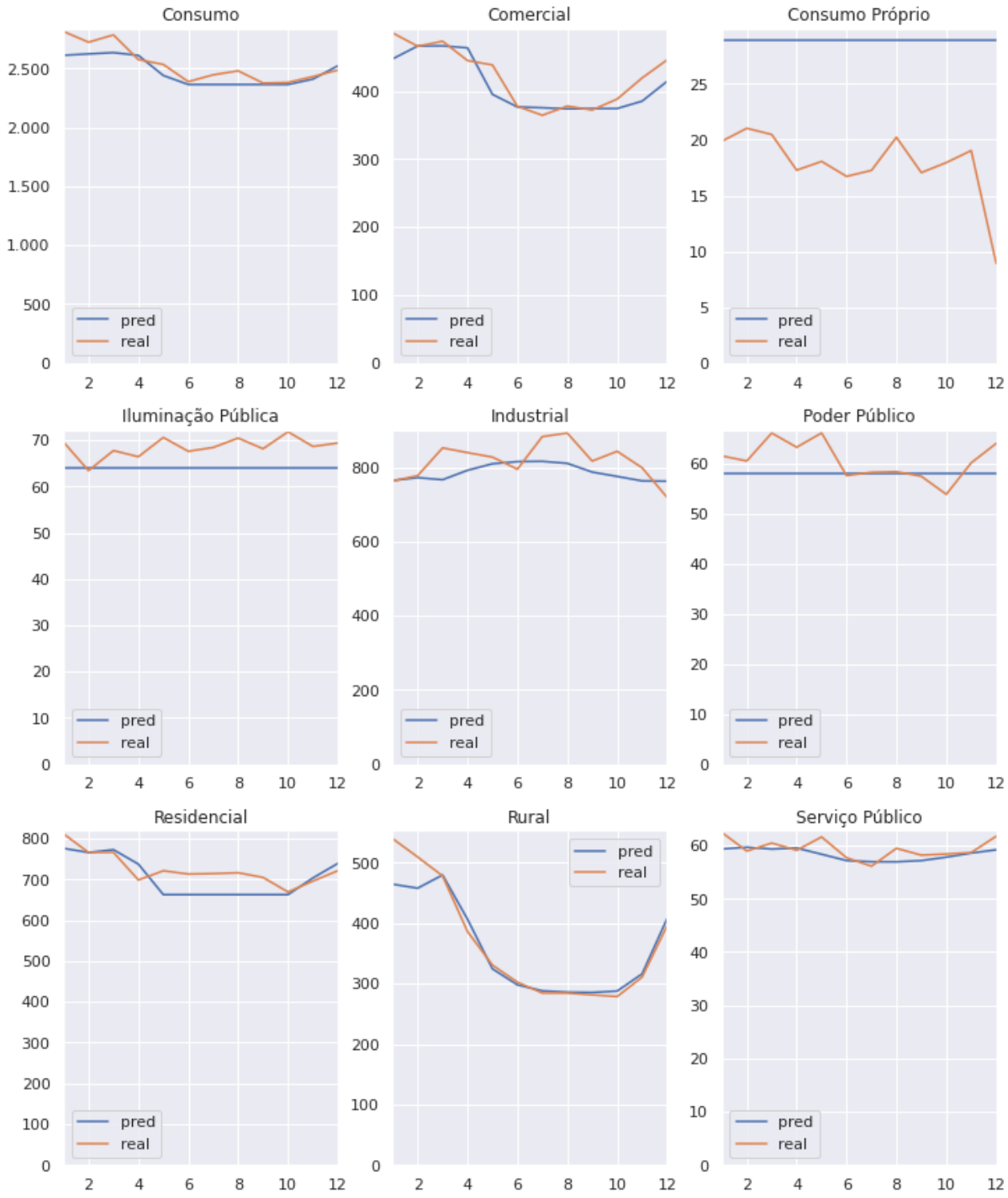
Na Figura 66 e na Figura 67 pode-se observar os valores preditos em comparação com os valores reais da série de 2018 para cada segmento e verifica-se que analisando cada setor ou a soma de todos separadamente, para este método, obtém-se uma resposta mais ajustada. E esse ganho de precisão, é devido a possibilidade de cada sub rede poder assumir um formato específico e tem tempos de parada distintos aproximando cada previsão ao melhor limite individual.

Figura 66 – Consumo mensal [kW] por SetorN1 (Previsão Interligada)



Fonte: Autor

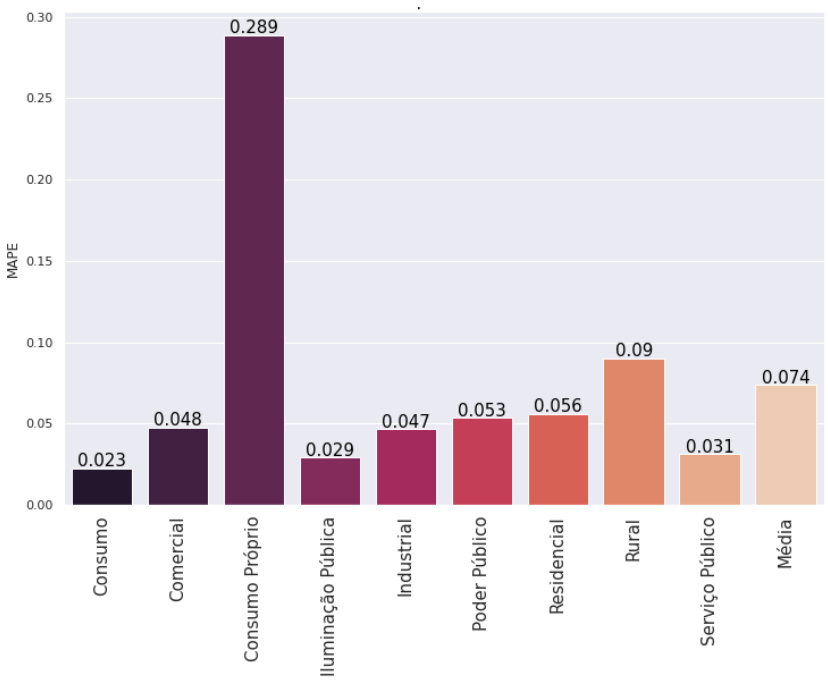
Figura 67 – Consumo mensal [kW] por SetorN1 (Previsão Paralela)



Fonte: Autor

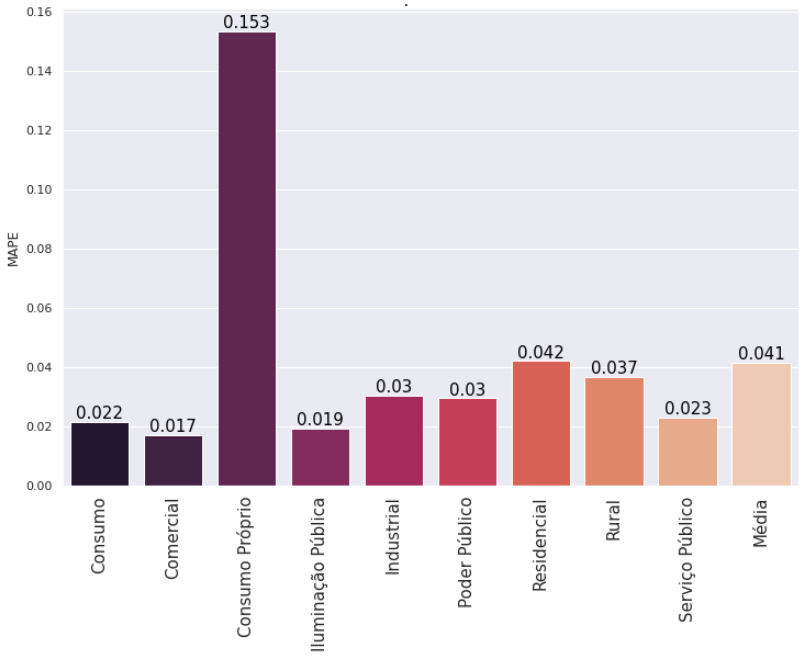
Avaliando os erros para cada segmento nas Figura 68 e Figura 69, é possível validar esta afirmação, de que uma rede para cada SetorN1 consegue uma melhor previsão do que uma única rede com todas os Setores para o método de ajuste discutido até o momento e com os hiper parâmetros e entradas utilizadas até este momento, cabe ainda ressaltar que apesar de obter precisões distintas nos dois testes, o pior caso se manteve, o que nos dá indícios de que ou as entradas escolhidas não sejam adequadas para prever o SetorN1 Consumo Próprio.

Figura 68 – Erro por SetorN1 (Previsão Interligada)



Fonte: Autor

Figura 69 – Erro por Setor N1(Previsão Paralela)



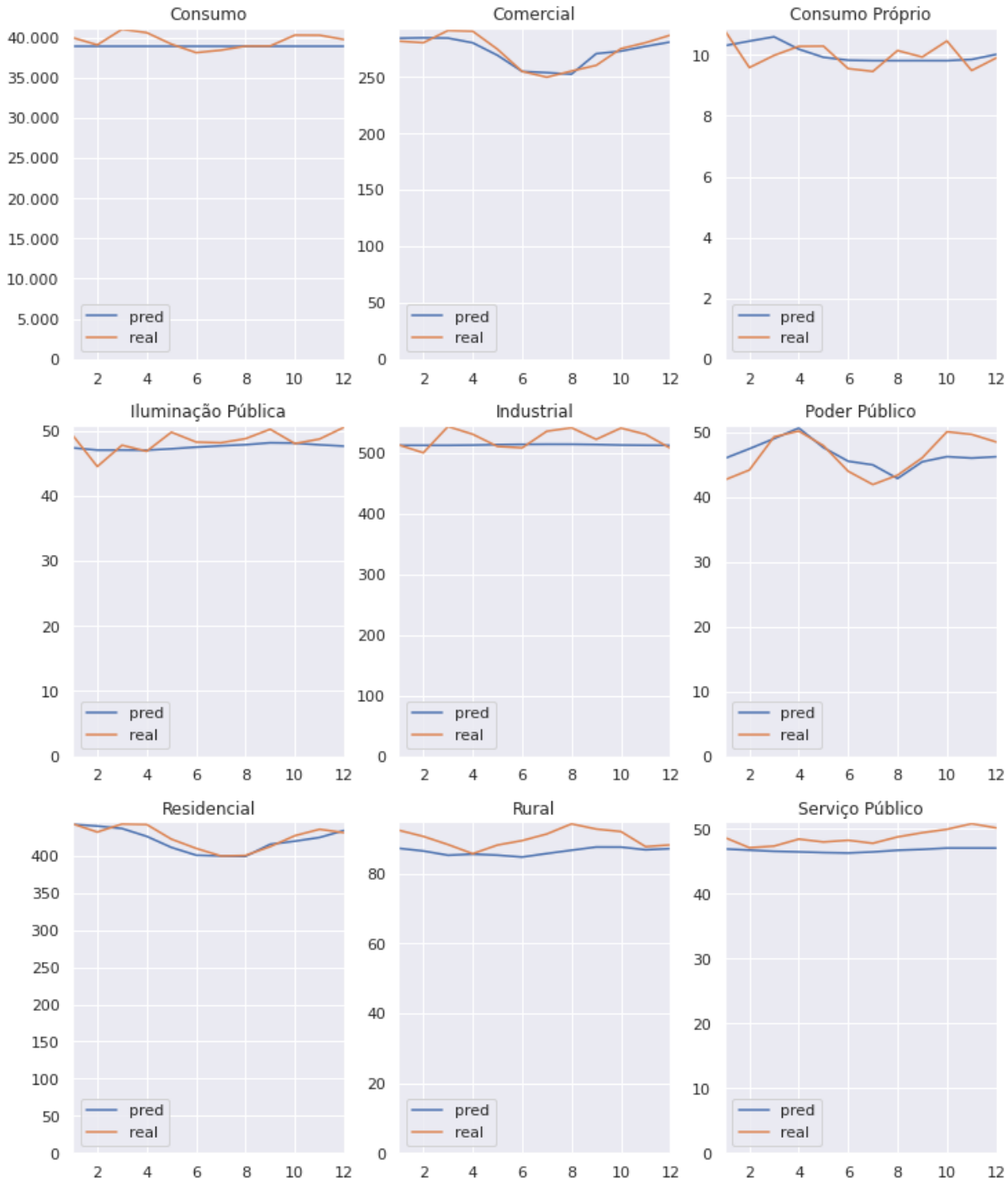
Fonte: Autor

3 ESTUDO DE CASO BR

Na generalização do método por unidade da federação ou para o Brasil considera-se que, a previsão do consumo de energia elétrica de um estado é similar a qualquer outro estado, e que o algoritmo genético proposto permite ajustar a rede da mesma forma que o fez para o estado do RS, segmentado por SetorN1 ou não. O modelo proposto tem suas estradas ajustadas para levar em conta todas as estações meteorológicas disponíveis no INMET e assumindo que as premissas utilizadas até este momento continuem válidas, é omitida a etapa de análise dos dados como foi realizado no estudo de caso RS. Assumindo que se o resultado do estudo de caso BR for válido então as premissas também são validas.

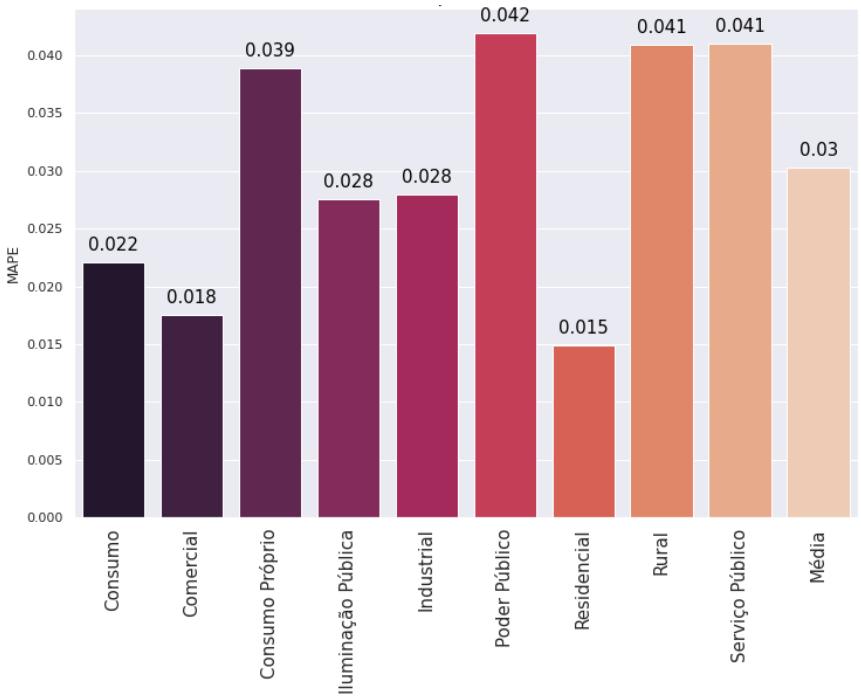
Na Figura 70 pode-se verificar que no ano de 2018 o consumo do Brasil não teve uma variabilidade tão grande quanto o estudo de caso RS, mas ainda assim o método obteve um bom ajuste dos valores preditos em comparação aos valores reais verificados. O que é comprovado na Figura 71, a qual traz os valores de erro dentro dos limites estipulados como meta para este estudo de $MAPE <= 5\%$.

Figura 70 - Teste Valor Real vs Predito LSTM BR



Fonte: Autor

Figura 71 – Erro por Setor N1 BR (Previsão Paralela)



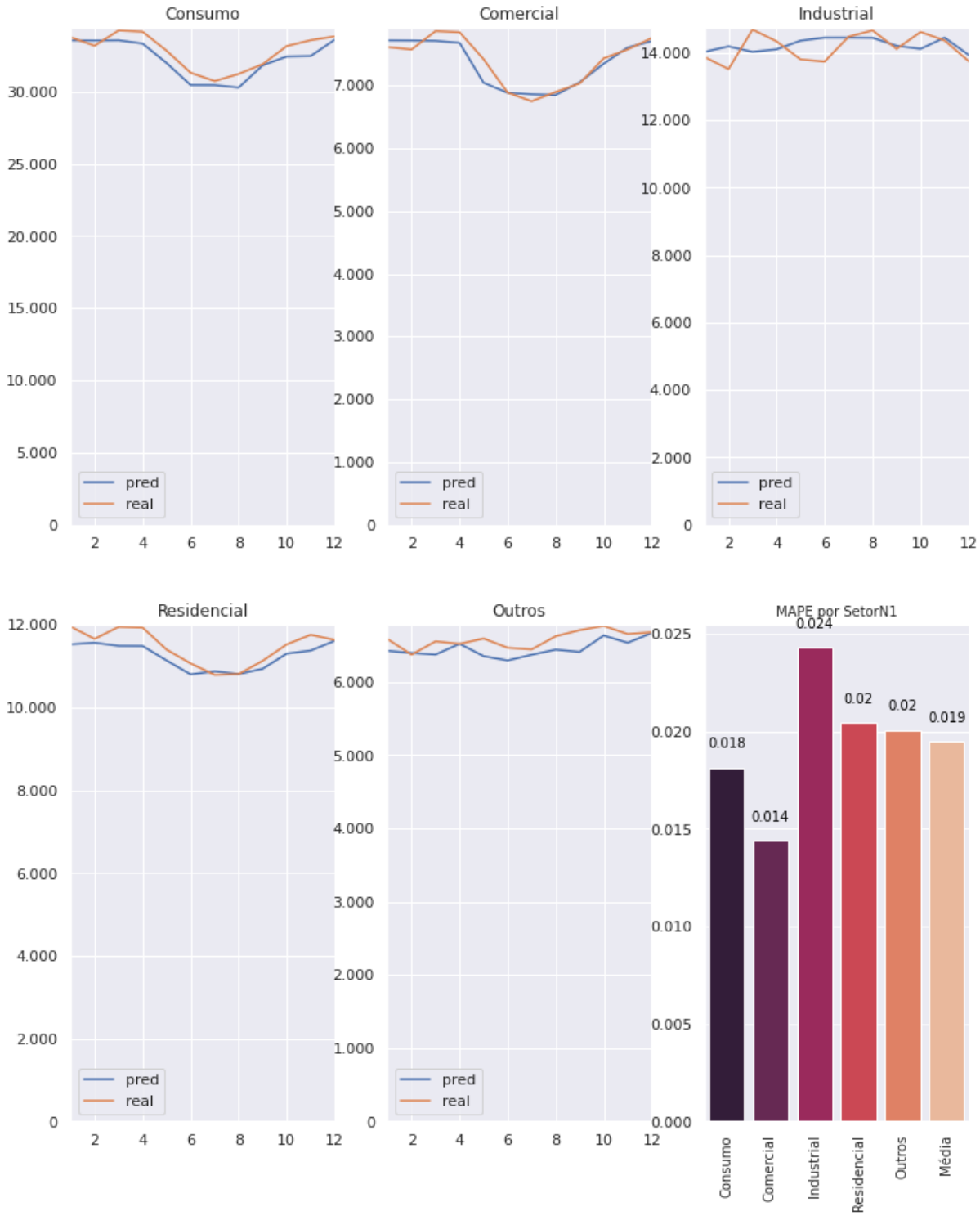
Fonte: Autor

3.1 Cenário de pandemia 2020

Ajustando-se aos dados disponíveis, até a data deste estudo, não estavam disponíveis os dados de do anuário EPE, publicação 2021 com os dados de 2020 como solução de contorno utilizou-se doravante os dados da “resenha mensal da EPE” [13], onde a decisão de usar esta referência foi indicação da própria EPE (em resposta a consulta por e-mail.), que são os dados preliminares que mais tarde irão compor o anuário com uma leve diferença, neste os dados são publicados, entretanto a resenha entrega os dados agrupados em quatro setores: Comercial, Industrial, Residencial e Outros. (A partir destes foi criado o “Consumo” para representar o consumo total)

Para avaliar o impacto destas novas variáveis na previsão foi realizado a previsão de carga com os dados de 2013 a 2017 prevendo 2018 (Figura 72). Para servir de parâmetro para as demais. E para garantir a repetibilidade e a comparação com os mesmos parâmetros para a definição da rede via algoritmo genético. Foi definida uma população inicial de 16 genomas e 32 épocas de iteração para cada classe de consumo.

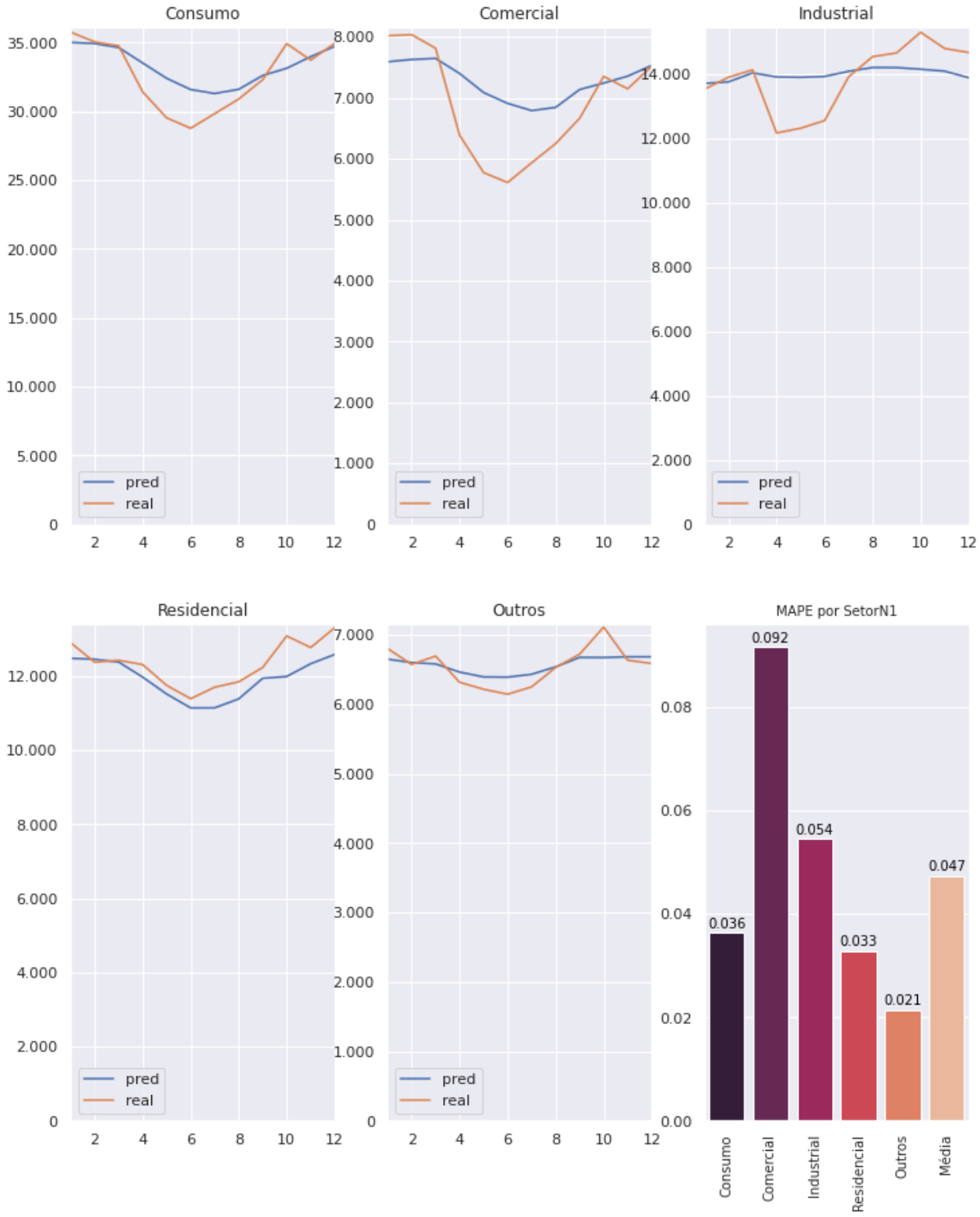
Figura 72 - Teste Valor Real vs Predito 2018 e MAPE (BR Resenha)



Fonte: Autor

No avanço da previsão para o ano de 2020 é possível verificar que todas as previsões não fornecem resultados adequados, como pode-se verificar na Figura 73.

Figura 73 - Teste Valor Real vs Predito 2020 e MAPE (BR Resenha)



Fonte: Autor

Com a evento da pandemia de COVID-19 houve impactos diretos na economia e por conseguinte no consumo de energia elétrica, no trabalho publicado pelo Banco Central do Brasil [19], as mortes por COVID foram correlacionadas a economia e ao indicador de atividade

econômica IBC-BR, já segundo o estudo sobre as mortes por covid EPICOVID [32] há no Brasil uma sub notificação de casos e óbitos por COVID-19.

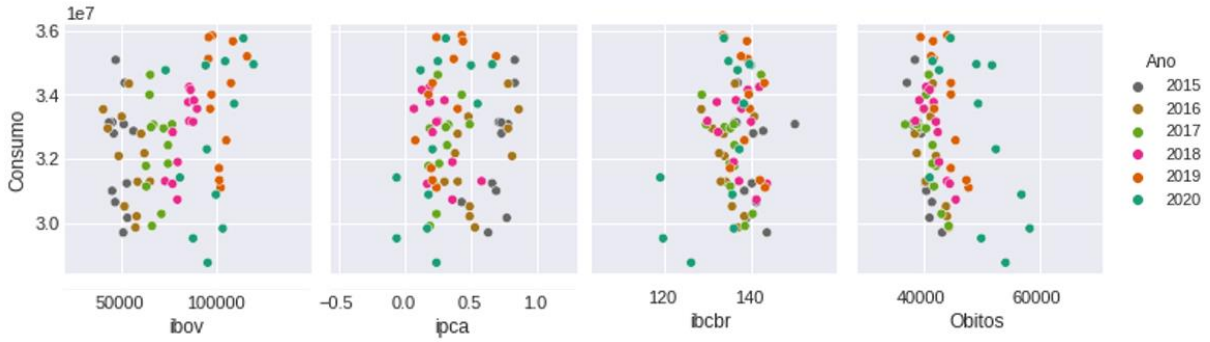
Assim uma vez que não há estatística previa, e considerando que outras doenças deixaram de ser tratadas no momento correto devido á pandemia, assumiu-se que o mais confiável seria utilizar o número de óbitos totais disponibilizados pela plataforma tabnet/DATASUS [17], apesar de não ser o número total de óbitos que estaria disponível no portal dos cartórios de registro civil [40], os dados fornecidos pelos mesmos não contemplam todo o período deste estudo, e assim, entende-se que os dados do SUS representam parcela suficiente significativa do todo.

Para melhor definir o cenário econômico, também foram agregadas como variáveis de entrada IPCA e IBOV, e estes escolhidos pela possibilidade de obter-se estudos de projeção oficiais, futuras pelo próprio Banco Central do Brasil e Bovespa.

Ao analisar variáveis duas a duas em um gráfico de dispersão cruzada, busca-se avaliar se a uma dependência de uma variável em relação a outra, ou seja, se um crescimento em um eixo representa o um crescimento ou decrescimento em outro, ainda ao contrastar esta dispersão para anos distintos busca-se identificar uma quebra de padrão ou segregação de um determinado período (meses ou anos) sinalizando um evento atípico ou anomalia.

Ao comparar a relação do consumo de energia elétrica no Brasil com as variáveis econômicas e sociais selecionadas, por esta análise, é possível observar uma segregação dos dados de 2020 (Figura 74) indicando uma clara anomalia entre o consumo e cada uma delas, corroborando a suposição inicial da escolha destas variáveis, em especial na variável óbitos.

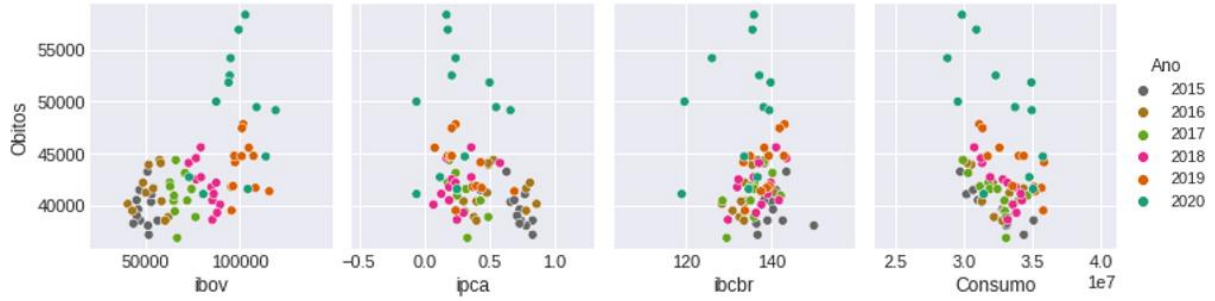
Figura 74 - Dispersão Cruzada: Consumo vs IBOV, IPCA, IBC-Br, Óbitos



Fonte: Autor

Colocando então sob análise principal a variável óbitos que apresentou maior segmentação na Figura 74, é possível verificar que o aumento no número de óbitos gerou impactos na bolsa de valores, na inflação, na atividade econômica e no consumo de energia elétrica, quebrando o padrão de todas estas variáveis (Figura 75) e justificando a escolha desta como principal variável em análise.

Figura 75 - Dispersão Cruzada: Óbitos vs Consumo vs IBOV, IPCA, IBC-Br, Consumo



Para garantir a correta escolha de variáveis foram realizados 16 testes de forma a testar todos os arranjos possíveis para as variáveis escolhidas e os resultados destes testes são mostrados no Quadro 1. No melhor arranjo testado (além das variáveis climáticas usadas em todos os cenários), foi obtido pelas entradas IBCBR, IPCA e Óbitos, é possível observar que estas variáveis estão presentes em 4 dos 5 melhores resultados.

Quadro 1 - Cenários de Teste

Entradas					MAPE						
Clima	IBOV	IBCBR	IPCA	Óbitos	Consumo	Comercial	Industrial	Residencial	Outros	Média	#
1		1	1	1	0,032	0,079	0,038	0,013	0,017	0,036	1
1	1	1	1	1	0,031	0,079	0,042	0,021	0,019	0,038	2
1	1	1		1	0,029	0,084	0,041	0,019	0,023	0,039	3
1		1	1		0,029	0,087	0,037	0,034	0,018	0,041	4
1	1		1	1	0,026	0,086	0,056	0,019	0,022	0,042	5
1				1	0,036	0,085	0,053	0,014	0,021	0,042	6
1	1	1			0,032	0,084	0,044	0,027	0,023	0,042	7
1		1		1	0,037	0,087	0,046	0,019	0,023	0,042	8
1	1	1	1		0,031	0,087	0,044	0,031	0,021	0,043	9
1			1	1	0,038	0,086	0,055	0,015	0,022	0,043	10
1	1				0,034	0,09	0,055	0,033	0,022	0,047	11
1					0,036	0,092	0,054	0,033	0,021	0,047	12
1		1			0,039	0,082	0,044	0,052	0,023	0,048	13
1			1		0,035	0,09	0,057	0,038	0,022	0,048	14
1	1		1		0,036	0,094	0,055	0,039	0,023	0,049	15
1	1			1	0,042	0,099	0,06	0,045	0,026	0,054	16

Pode-se evidenciar na Figura 76 (que ilustra o melhor arranjo testado) quando comparado a Figura 73, houve melhora na previsão de todas as classes e que a configuração considerando apenas o clima, ou seja a 12ª posição, seria a melhor previsão dentre as 16 testadas, também é possível verificar no Quadro 1, que a inclusão da variável óbitos melhora a previsão da 12ª para 6ª posição.

Figura 76 - Teste Valor Real vs Predito 2020 e MAPE (BR Resenha IBCBR, IPCA, Óbitos)

Fonte: Autor

Reaplicando esta mesma estrutura para a previsão de 2018 a fim de verificar se as novas variáveis interfeririam positivamente também num cenário sem pandemia. Pode-se verificar na Figura 77 teve um desempenho levemente superior quando comparados os erros (MAPE) da Figura 72, demonstrando que o uso destas novas variáveis é mais relevante em período de pandemia do que fora dele.

Figura 77 - Teste Valor Real vs Predito 2018 e MAPE (BR Resenha IBCBR, IPCA, Óbitos)

Fonte: Autor

Os demais resultados estão detalhados no Anexo 7 – Resultados dos 16 cenários de teste com diferentes arranjos de entrada.

4 CONSIDERAÇÕES FINAIS

4.1 Conclusões

A análise das aplicações do modelo aqui proposto comprovou que a técnica escolhida, regressor LSTM/MLP, associada a um dimensionamento da rede neural através de algoritmo genético, foi eficiente na previsão do consumo de energia elétrica. A inclusão da variável do número de óbitos, associada as tradicionais variáveis econômicas e sociais, mesmo em um contexto adverso como a pandemia de COVID-19, resultou em melhoria na previsão de consumo de energia elétrica.

Atendendo a regulamentação da Agência Nacional de Energia Elétrica - ANEEL quanto ao erro de projeção, o modelo permite obter uma previsão com erro menor do que 5%, resultando em um erro médio final de 3,6%, a exceção foi a classe comercial que apresentou um erro de 7,9% em ambiente pandêmico, devido à falta de uma variável sócio-econômica que se correlacionasse com esta classe. No entanto, em um contexto fora do período de pandemia, obteve-se um erro médio de 1,6%, e um melhor caso de erro de 1,1% para a classe “Outros” e um pior caso de 2,1% para a classe “Consumo Total”, ambos os resultados são considerados muito bons.

O modelo fornece projeções de consumo de energia elétrica relevantes para o planejamento de sistemas elétricos e, como consequência, o melhor uso dos recursos energéticos, com o uso de variáveis e informações públicas de fácil acesso, no que se refere a dados históricos e projeções.

As contribuições do trabalho são a comprovação de que o número de óbitos influencia o consumo de energia e que um enfoque socio-econômico-ambiental permite uma melhor previsão de consumo. Ainda, a estrutura do modelo algorítmico ajusta via algoritmo de otimização genética uma rede neural a partir de quaisquer entradas numéricas e assim pode estimar um valor de saída com base em dados históricos e projeções correlatas, desde que sejam testadas e pesquisadas entradas adequadas para cada estudo de caso.

O trabalho aqui desenvolvido revelou aperfeiçoamentos no modelo proposto, como, por exemplo, o estudo do número de internações que não resultaram em óbitos e de um indicador econômico que melhor caracterize a atividade comercial. A comparação deste método com métodos usualmente empregados nos estudos de projeção de consumo de energia elétrica também é desejável, visando melhoramentos.

4.2 *Trabalho a ser publicado*

Um resultado desse Projeto de Diplomação é o artigo intitulado “Modelo de previsão de consumo de energia elétrica em cenário pandêmico com uso de ferramentas *machine learning”*, no qual é descrito o modelo desenvolvido e os principais resultados de sua aplicação. O artigo será apresentado e publicado em maio de 2022 no XXVI Seminário Nacional de Produção e Transmissão de Energia Elétrica – SNPTEE.

REFERÊNCIAS

[1] G. v. Rossum, “Python,” [Online]. Available: https://www.python.org/. [Acesso em 2021].

[2] Empresa de Pesquisa Energética, *Metodologia: Modelo de Projeção da Demanda de Eletricidade*, Rio de Janeiro, RJ: Ministério de Minas e Energia, 2019, p. 33.

[3] C. Jaipradidtham, “Next Day Load Demand Forecasting of Future in Electrical Power Generation on Distribution Networks using Adaptive Neuro-Fuzzy Inference,” *2006 IEEE International Power and Energy Conference*, pp. 64-67, 2006.

[4] M. A. M. T. a. H. V. P. M. Ali, “Load Forecasting Through Estimated Parametrized Based Fuzzy Inference System in Smart Grids,” *in IEEE Transactions on Fuzzy Systems*, vol. 29, pp. 156-165, Janeiro 2021.

[5] A. S. A.-S. M. K. E. M. A. a. A. K.-F. M. A. Mohamed, “A Reliability-Oriented Fuzzy Stochastic Framework in Automated Distribution Grids to Allocate μ-PMUs,” *in IEEE Access*, vol. 7, pp. 33393-33404, 2019.

[6] Y. S. L. W. J. L. L. Z. a. S. W. Q. Liu, “A hybrid FCW-EMD and KF-BA-SVM based model for short-term load forecasting,” *in CSEE Journal of Power and Energy Systems*, vol. 4, nº 2, pp. 226-237, Junho 2018.

[7] H. H. a. W. Y. W. Xu, “Energy Time Series Forecasting Based on Empirical Mode Decomposition and FRBF-AR Model,” *in IEEE Access*, vol. 7, pp. 36540-36548, 2019.

[8] N.-A.-M. S. R. D. a. E. H. S. H. Rafi, “A Short-Term Load Forecasting Method Using Integrated CNN and LSTM Network,” *IEEE Access*, 2021.

[9] A. N. d. E. E. (Brasil), *Resolução Normativa 414/2010: atualizada até a REN499/2012 / Agência Nacional de Energia Elétrica*, Brasilia, 2012, p. 212.

[10] EPE - Empresa de Pesquisa Energética, 2021. [Online]. Available: https://www.epe.gov.br/. [Acesso em janeiro 2021].

[11] Empresa de Pesquisa Energética, “EPE,” EPE, 2019. [Online]. Available: https://www.epe.gov.br/pt/publicacoes-dados-abertos/dados-abertos/dados-do-anuario-estatistico-de-energia-eletrica. [Acesso em Março 2021].

[12] Empresa de Pesquisa Energética, “Plano Decenal de Expansão de Energia 2029,” Ministério de Minas e Energia, Brasília, 2020.

[13] Empresa de Pesquisa Energética, “EPE - Resenha Mensal,” [Online]. Available: https://www.epe.gov.br/pt/publicacoes-dados-abertos/publicacoes/resenha-mensal-do-mercado-de-energia-eletrica. [Acesso em julho 2021].

[14] INMET - Instituto Nacional de Meteorologia, 2021. [Online]. Available: mapas.inmet.gov.br. [Acesso em Janeiro 2021].

[15] Á. Justen, “dataset: dataset covid19,” Brasil IO, 22 07 2021. [Online]. Available: https://brasil.io/dataset/covid19/files/. [Acesso em 22 07 2021].

[16] ARPENBrasil, “Portal Transparência Registro Civil,” ARPEN Brasil, [Online]. Available: https://transparencia.registrocivil.org.br/especial-covid. [Acesso em 21 Julho 2021].

[17] Ministério da Saúde, “DataSus,” [Online]. Available: https://datasus.saude.gov.br/. [Acesso em mar 2021].

[18] Banco Central do Brasil, “Dados Abertos,” [Online]. Available: https://dadosabertos.bcb.gov.br/dataset/. [Acesso em ago 2021].

[19] B. C. d. B. -. BCB, “Intensidade da pandemia e atividade econômica,” Junho 2021. [Online]. Available: https://www.bcb.gov.br/content/ri/relatorioinflacao/202106/ri202106b1p.pdf. [Acesso em 22 Julho 2021].

[20] B3 - Brasil Bolsa Balcão, “Market Data e Indices,” [Online]. Available: http://www.b3.com.br/pt_br/market-data-e-indices/indices/indices-

amplos/indice-ibovespa-ibovespa-estatisticas-historicas.htm. [Acesso em jun 2021].

[21] IEEE Spectrum, 2020. [Online]. Available: https://spectrum.ieee.org/top-programming-languages/. [Acesso em abril 2020].

[22] Google, “Google Colaboratory,” 2021. [Online]. Available: colab.research.google.com.

[23] J. Hunte, D. Dale, E. Firing e M. Droettboom, “MatplotLib,” [Online]. Available: matplotlib.org. [Acesso em 2021].

[24] “Numpy,” 2021. [Online]. Available: https://numpy.org/.

[25] “Pandas,” 2021. [Online]. Available: pandas.pydata.org.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. ubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot e E. Duchesnay, *scikit-learn: Machine Learning in Python,* vol. 12, 2011, pp. 2825-2830.

[27] “Seaborn,” [Online]. Available: seaborn.pydata.org. [Acesso em Março 2021].

[28] S. Seabold e J. Perktold, “statsmodels: Econometric and statistical modeling with python,” *9th Python in Science Conference,* 2010.

[29] A. M.Abadi, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015.

[30] CPTEC - Centro de Previsão de Tempo e Estudos Climáticos, 2021. [Online]. Available: http://enos.cptec.inpe.br/. [Acesso em janeiro 2021].

[31] MSF - Médicos sem Fronteiras, 2021. [Online]. Available: https://coronavirus.msf.org.br/o-que-e-covid-19/. [Acesso em janeiro 2021].

[32] H. P. M. A. B. A. H. B. B. F. Mesenburg MA, *Chronic non-communicable diseases and COVID-19: EPICOVID-19 Brazil results,* vol. 55, 2021, p. 38.

[33] F. ROSENBLATT, “THE PERCEPTRON: A PROBABILISTIC MODEL FOR,” *Psychological Review,* vol. 65, nº 6, pp. 386-408, 23 abril 1958.

[34] A. Géron, Mãos à Obra Aprendizado de Máquina com Scikit-Learn & TensorFlow: Conceitos, Ferramentas e Técnicas Para a Construção de Sistemas Inteligentes, Rio de Janeiro: Alta Books, 2019.

[35] Asimov Institute, 2016. [Online]. Available: https://www.asimovinstitute.org/. [Acesso em janeiro 2021].

[36] M. Schuster e K. K. Paliwal, “Bidirectional Recurrent Neural Networks,” *IEEE Transactions on Signal Processing,* vol. 45, pp. 2673-2681, 1997.

[37] D. P. K. a. J. Ba, “Adam: A Method for Stochastic Optimization,” *3rd International Conference for Learning Representations,* nº v9, 2015.

[38] L. D. Chambers, The practical handbook of genetic algorithms, applications, Boca Raton, Florida: Chapman & Hall/CRC, 2001, p. 544.

[39] R. Reguant, “Roulette Wheel Selection Python,” 2019. [Online]. Available: https://rocreguant.com/roulette-wheel-selection-python/2019/. [Acesso em março 2021].

[40] Transparência, “Registro Civil,” [Online]. Available: https://transparencia.registrocivil.org.br/registros. [Acesso em abril 2021].

[41] BIBENG, “Manual de Normalização de Trabalhos Acadêmicos,” 2019. [Online]. Available: https://www.ufrgs.br/bibeng/wp-content/uploads/Manual-de-normaliza%C3%A7%C3%A3o-setembro-2019.pdf. [Acesso em 28 04 2020].

[42] H. A. GIL e A. M. GUITIÉRREZ, *Modelo de Previsão do Consumo de Energia Elétrica da Classe Industrial do Brasil.,* Universidade Federal de Santa Catarina, 1997.

[43] M. Harrison, *Machine Learning – Guia de Referência Rápida: Trabalhando com dados estruturados em Python,* 1ed ed., São Paulo, SP: Novatec Editora, 2020.

[44] C. P. d. A. e. A. d. M. d. E. E. (COPAM), “Dados do Anuário Estatístico de Energia Elétrica,” Empresa de Pesquisa Energética - EPE, 2019. [Online]. Available: https://epe.gov.br/pt/publicacoes-dados-abertos/dados-abertos/dados-do-anuario-estatistico-de-energia-eletrica. [Acesso em 23 junho 2020].

[45] U. R. N. a. R. Costa-Castelló, “A Model Predictive Control-Based Energy Management Scheme for Hybrid Storage System in Islanded Microgrids,” *in IEEE Access*, vol. 8, pp. 97809-97822, 2020.

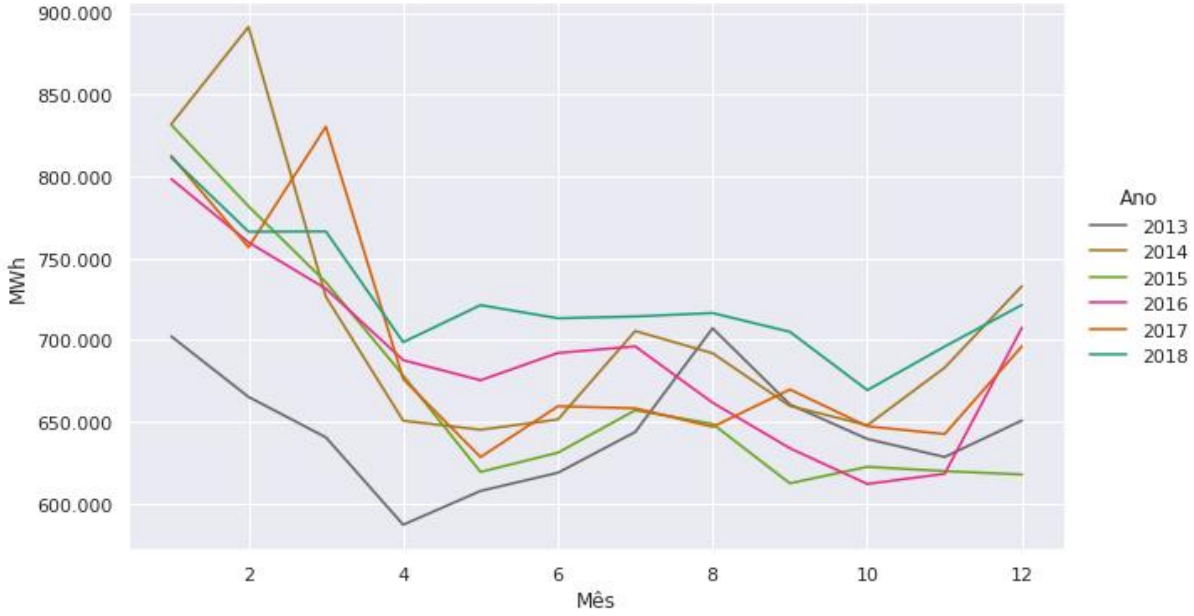
[46] Empresa de Pesquisa Energética, *Projeção da Demanda de Energia Elétrica para os próximos 10 anos (2017-2026)),* Rio de Janeiro, RJ: Ministério de Minas e Energia, 2017.

ANEXOS

Anexo 1 – Análise de Consumo Mensal por Setor N1 no RS

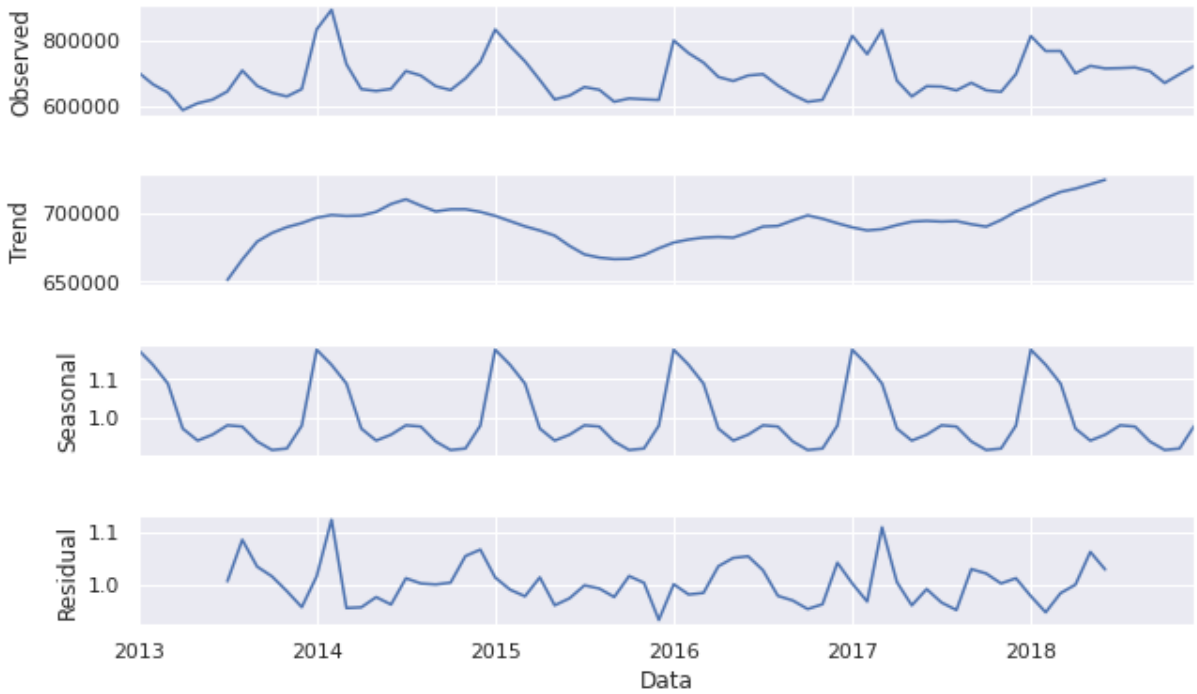
Residencial

Figura 78 - Perfil de Consumo Mensal, Detalhe: Setor N1 = Residencial, UF = RS



Fonte: Autor

Figura 79 - Decomposição Sazonal Multiplicativa do Perfil de Consumo,Detalhe: Setor N1 = Residencial e UF = RS



Fonte: Autor

Figura 80 – Autocorrelação, Detalhe: Setor N1 = Residencial e UF = RS

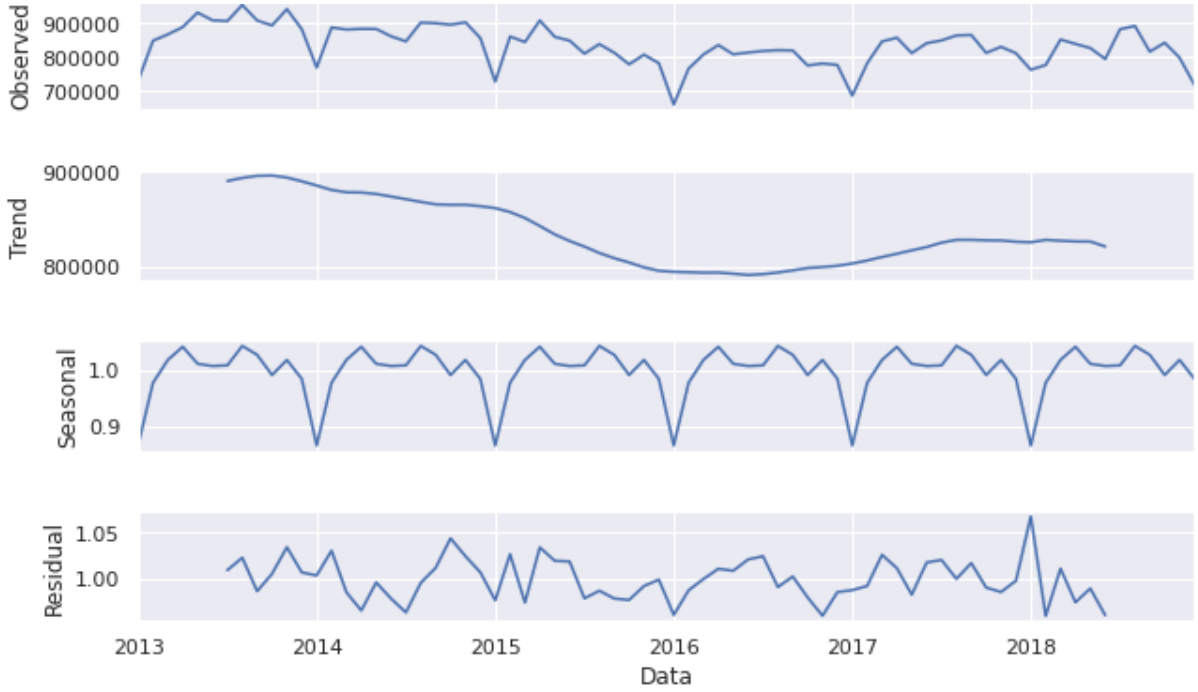
Fonte: Autor

Industrial

Figura 81 - Perfil de Consumo Mensal, Detalhe: Setor N1 =Industrial, UF = RS

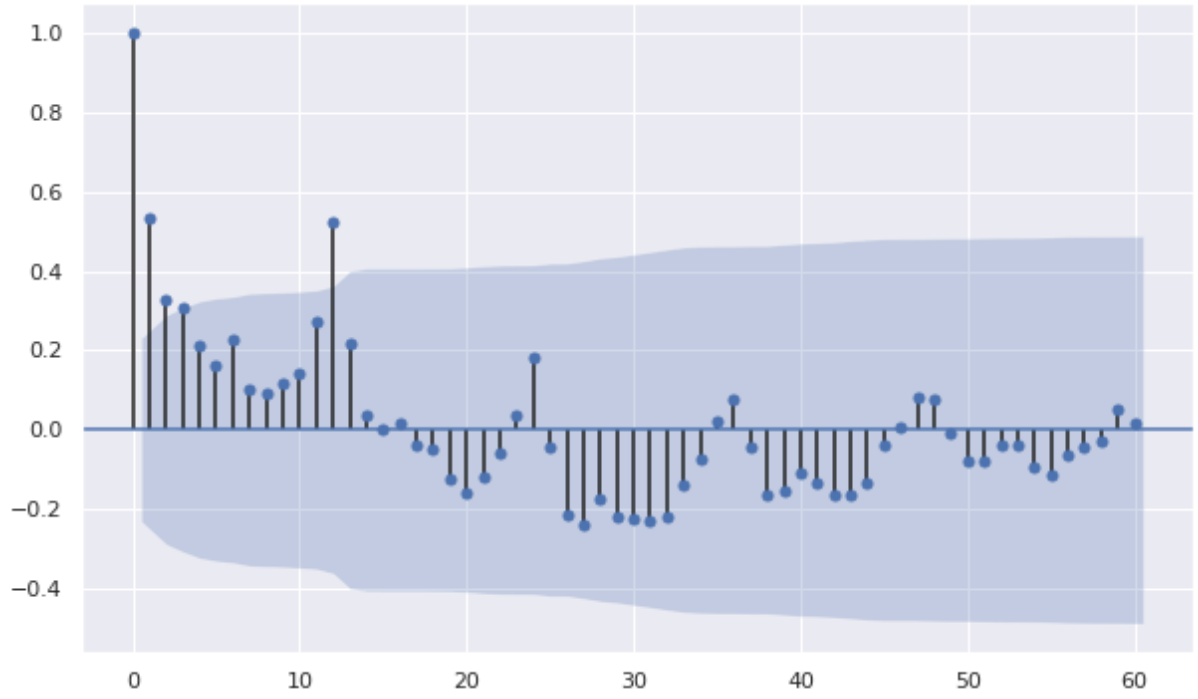
Fonte: Autor

Figura 82 - Decomposição Sazonal Multiplicativa do Perfil de Consumo, Detalhe: Setor N1 = Industrial e UF = RS



Fonte: Autor

Figura 83 – Autocorrelação, Detalhe: Setor N1 = Industrial e UF = RS



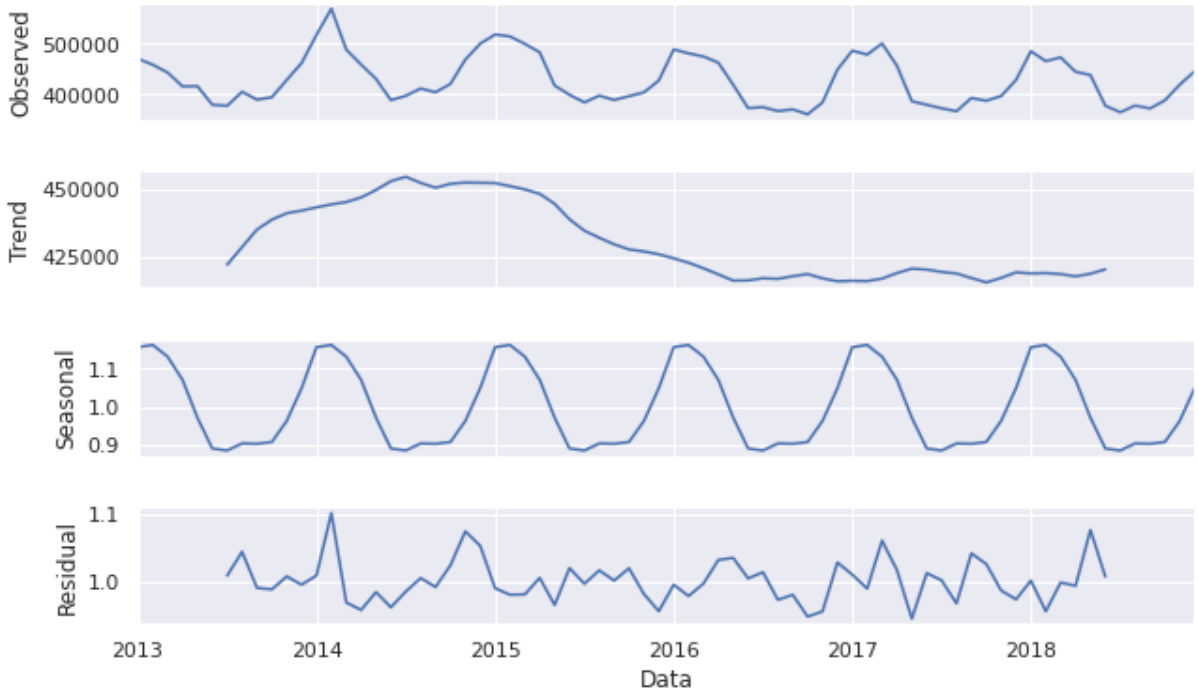
Fonte: Autor

Comercial

Figura 84 - Perfil de Consumo Mensal, Detalhe: Setor N1 = Comercial, UF = RS

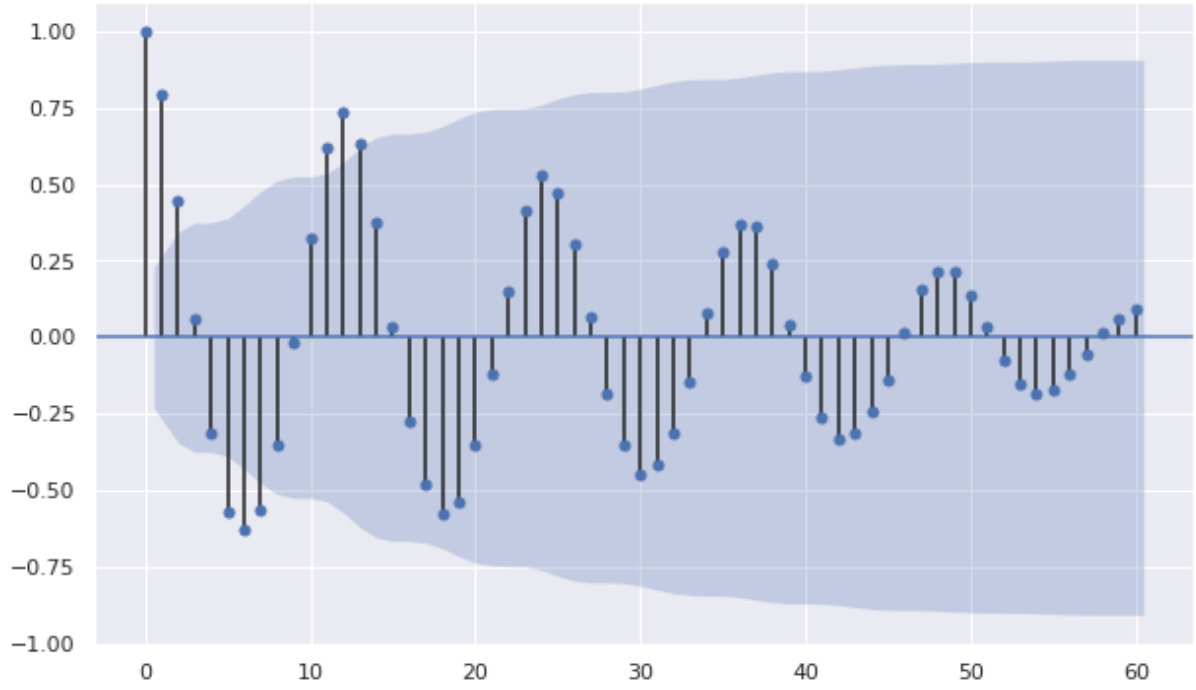
Fonte: Autor

Figura 85 - Decomposição Sazonal Multiplicativa do Perfil de Consumo, Detalhe: Setor N1 = Comercial e UF = RS



Fonte: Autor

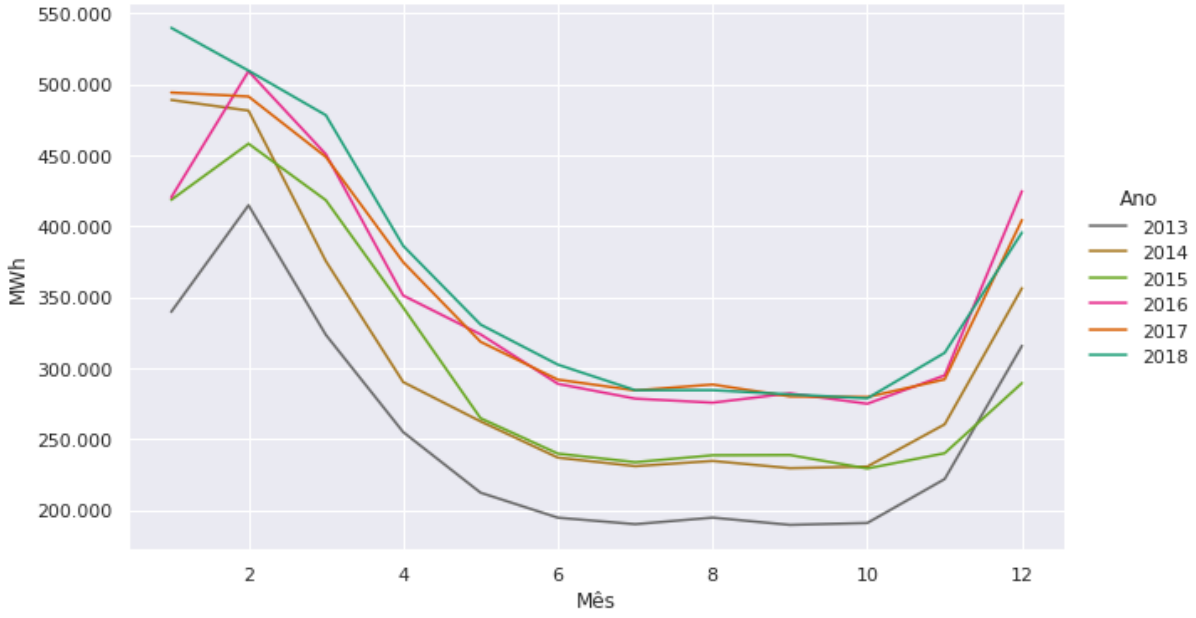
Figura 86 – Autocorrelação, Detalhe: Setor N1 = Comercial e UF = RS



Fonte: Autor

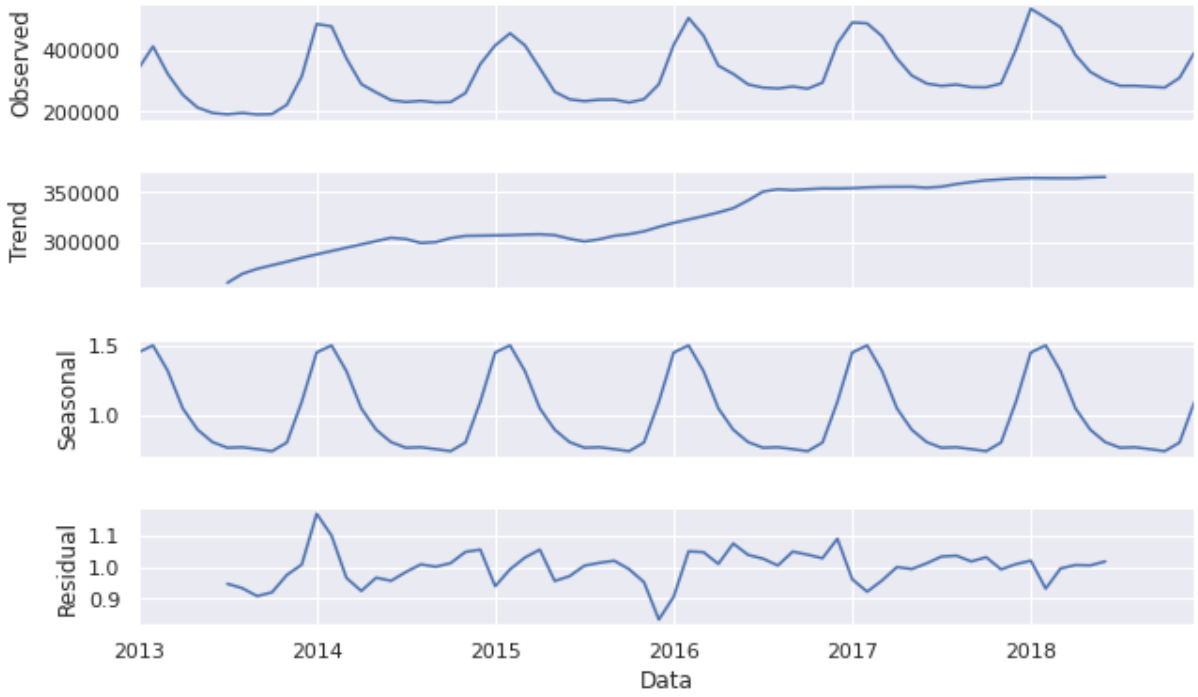
Rural

Figura 87 - Perfil de Consumo Mensal, Detalhe: Setor N1 =Rural, UF = RS



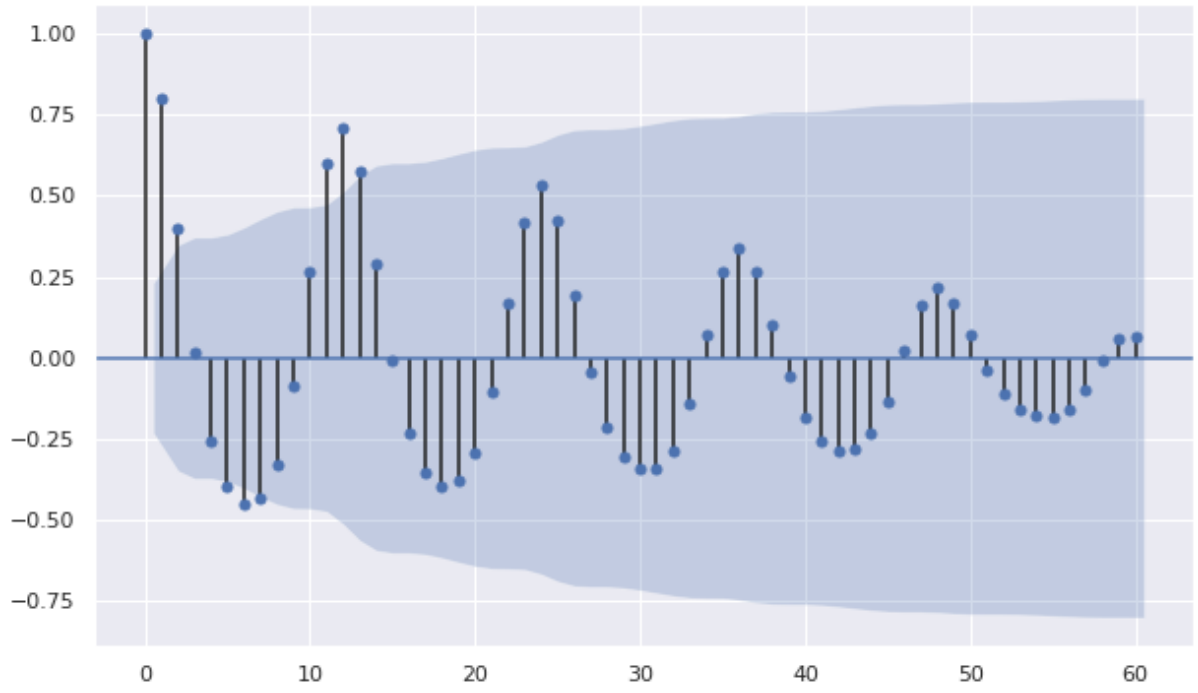
Fonte: Autor

Figura 88 - Decomposição Sazonal Multiplicativa do Perfil de Consumo,Detalhe: Setor N1 = Rural e UF = RS



Fonte: Autor

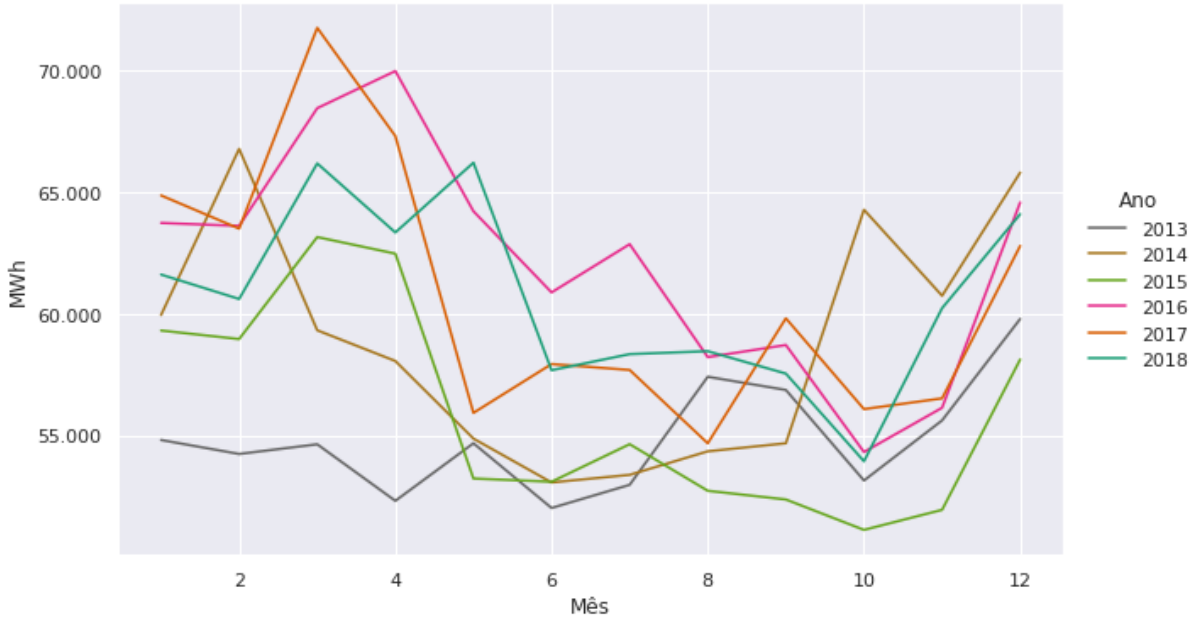
Figura 89 – Autocorrelação, Detalhe: Setor N1 = Rural e UF = RS



Fonte: Autor

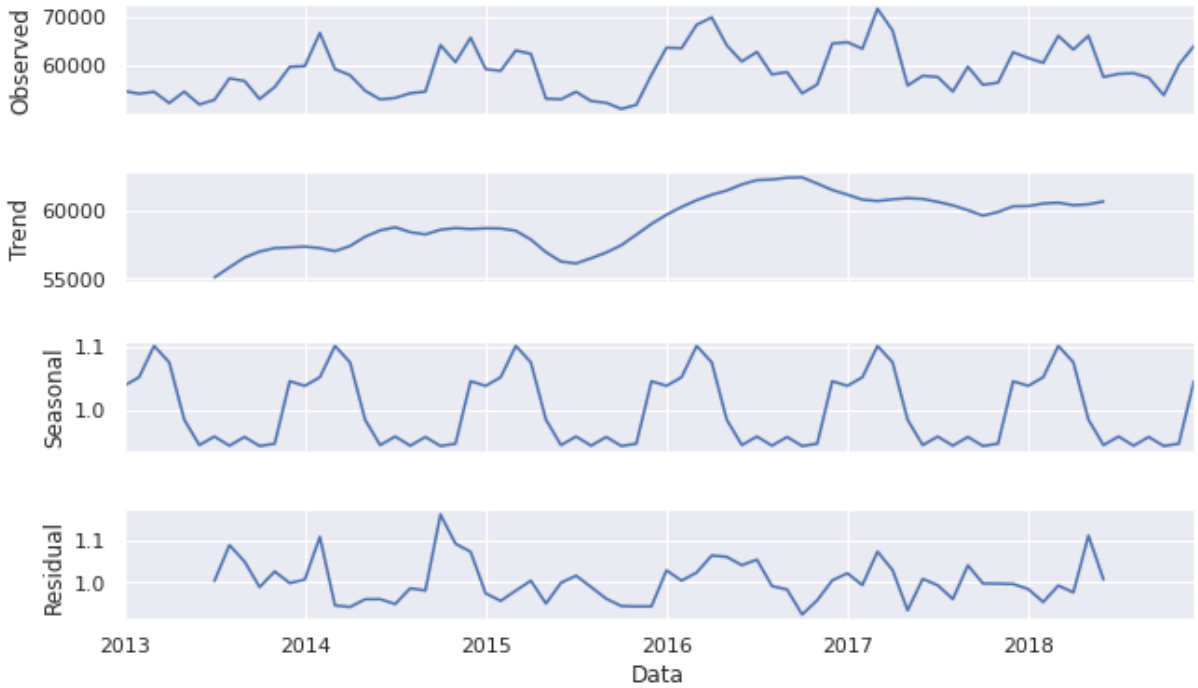
Poder Público

Figura 90 - Perfil de Consumo Mensal, Detalhe: Setor N1 = Poder Público, UF = RS



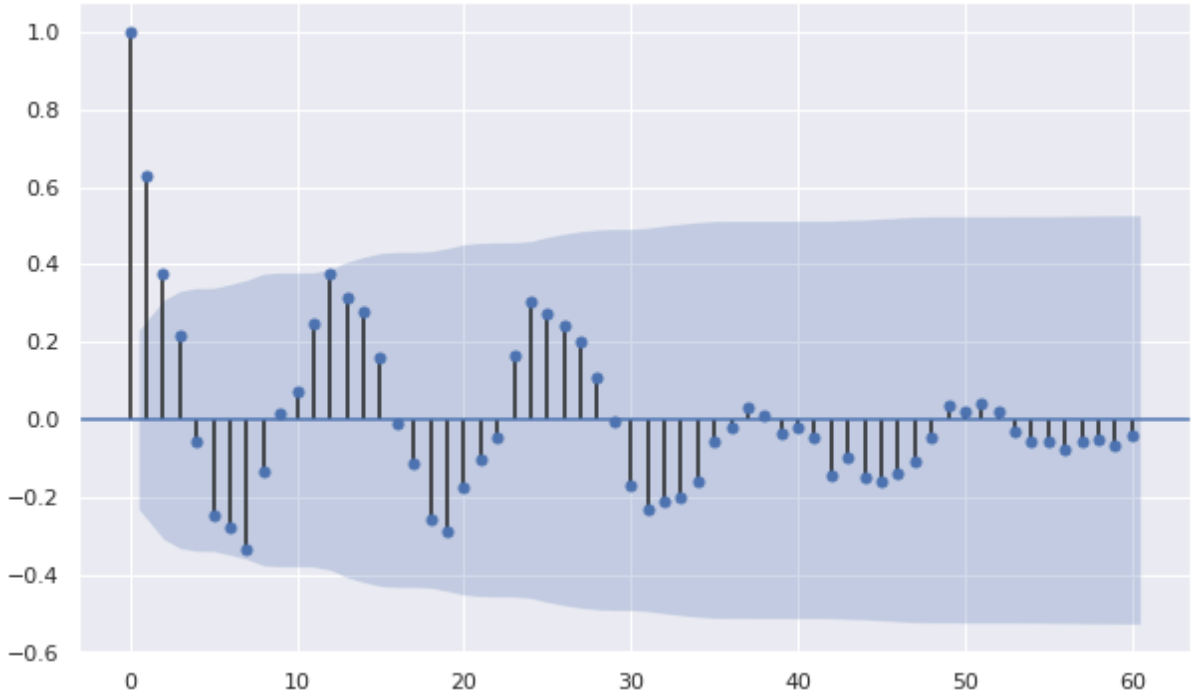
Fonte: Autor

Figura 91 - Decomposição Sazonal Multiplicativa do Perfil de Consumo, Detalhe: Setor N1 = Poder Público e UF = RS



Fonte: Autor

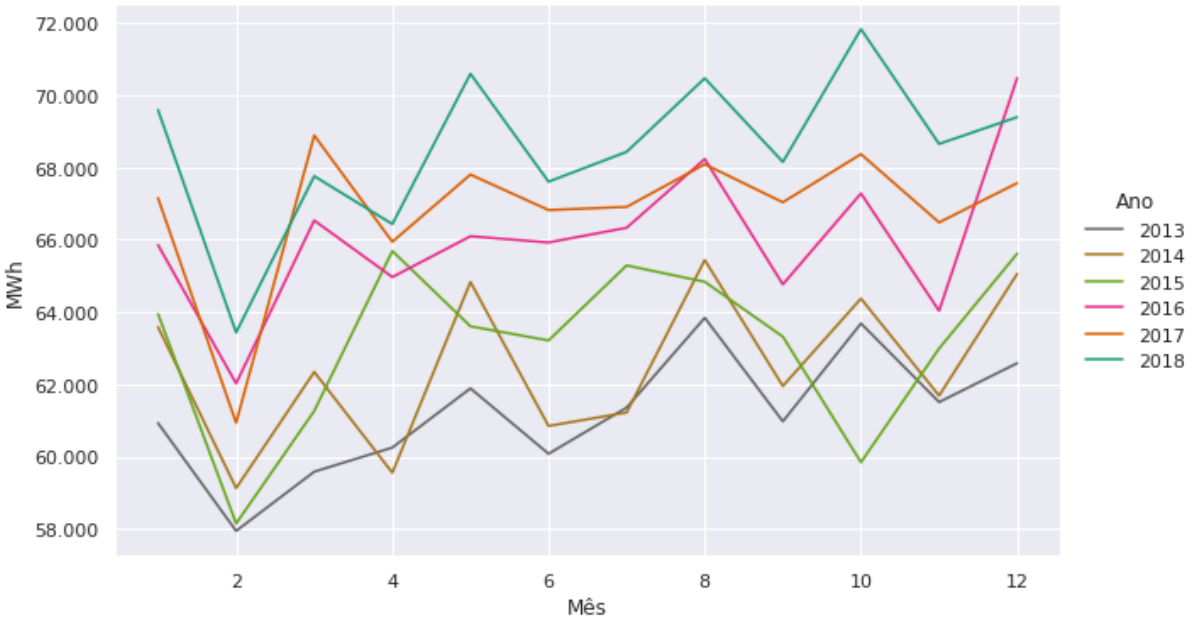
Figura 92 – Autocorrelação, Detalhe: Setor N1 = Poder Público e UF = RS



Fonte: Autor

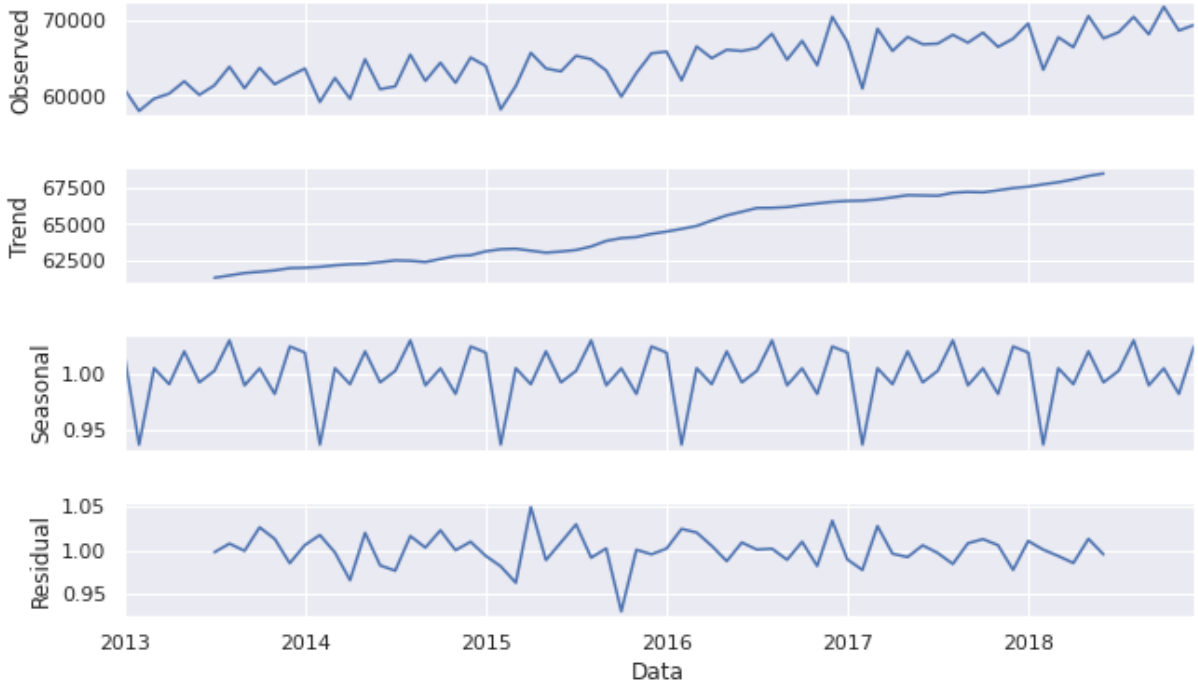
Iluminação Pública

Figura 93 - Perfil de Consumo Mensal, Detalhe: Setor N1 = Iluminação Pública, UF = RS



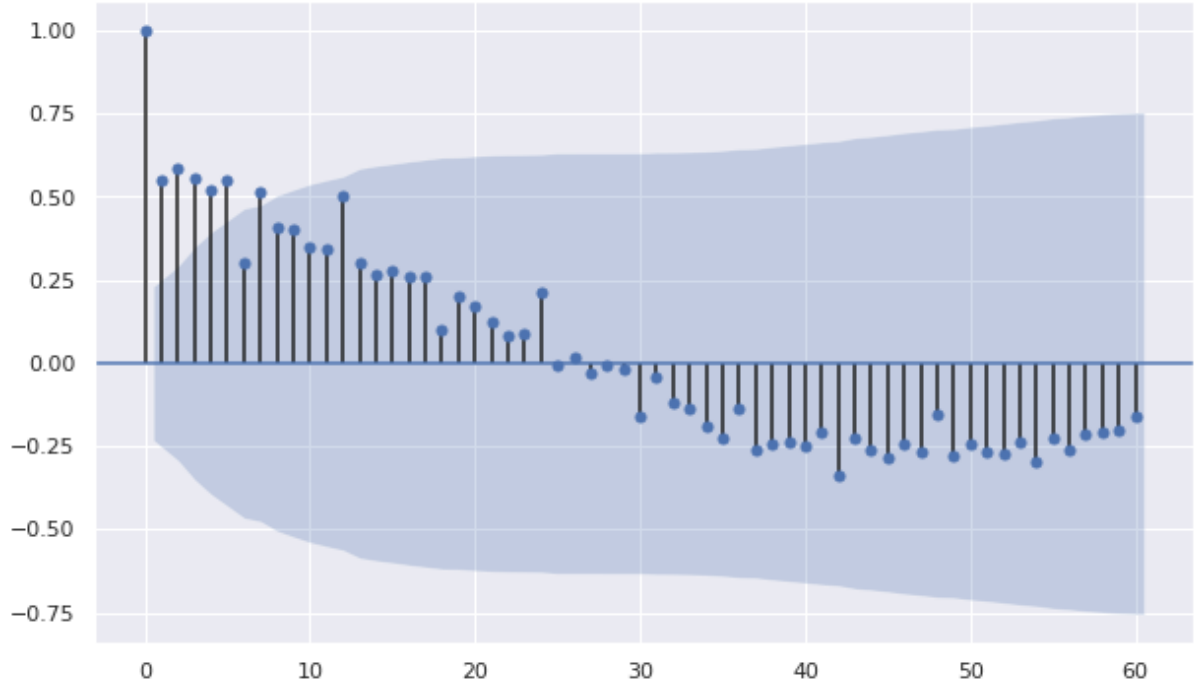
Fonte: Autor

Figura 94 - Decomposição Sazonal Multiplicativa do Perfil de Consumo, Detalhe: Setor N1 = Iluminação Pública e UF = RS



Fonte: Autor

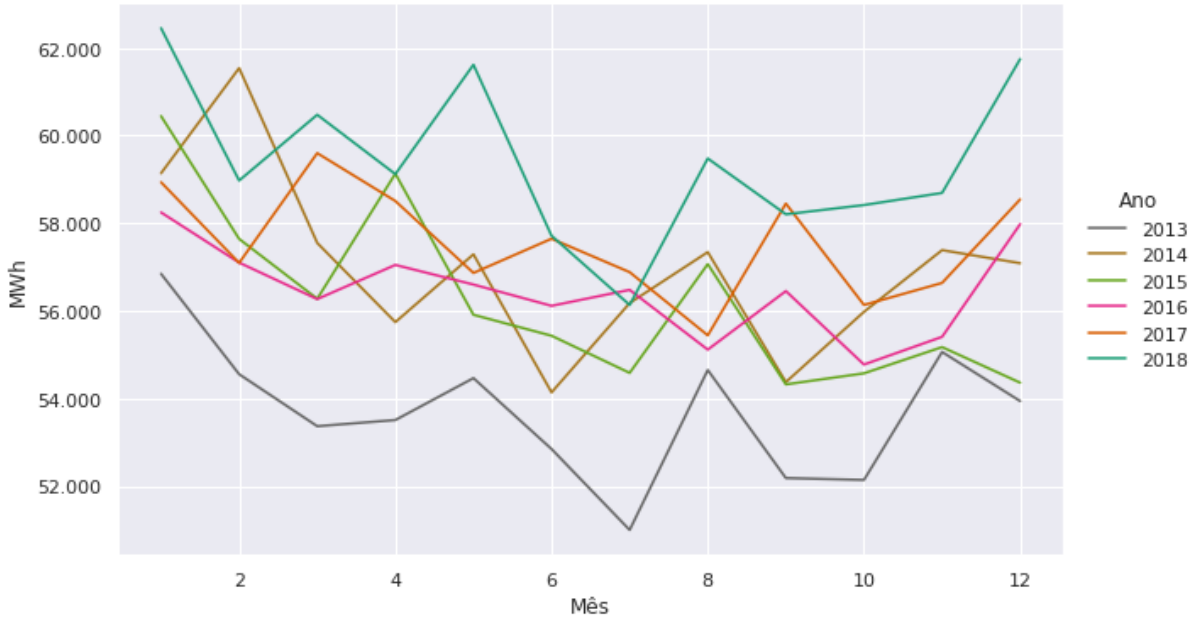
Figura 95 – Autocorrelação, Detalhe: Setor N1 = Iluminação Pública e UF = RS



Fonte: Autor

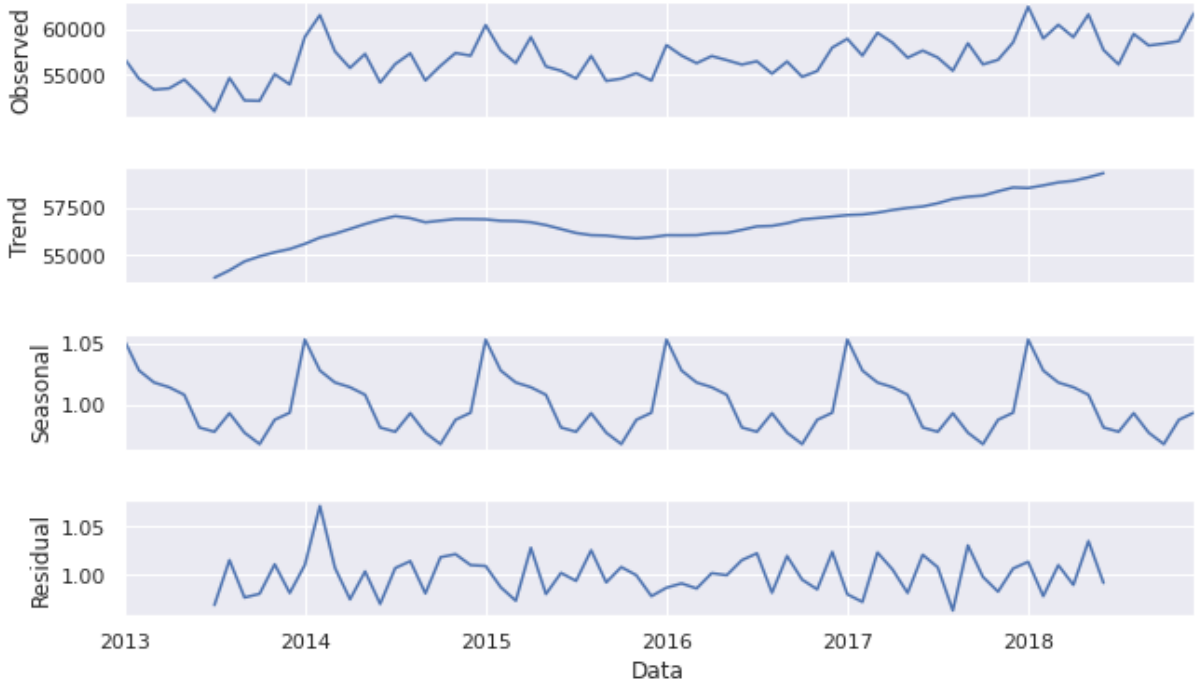
Serviço Público

Figura 96 - Perfil de Consumo Mensal, Detalhe: Setor N1 = Serviço Público, UF = RS



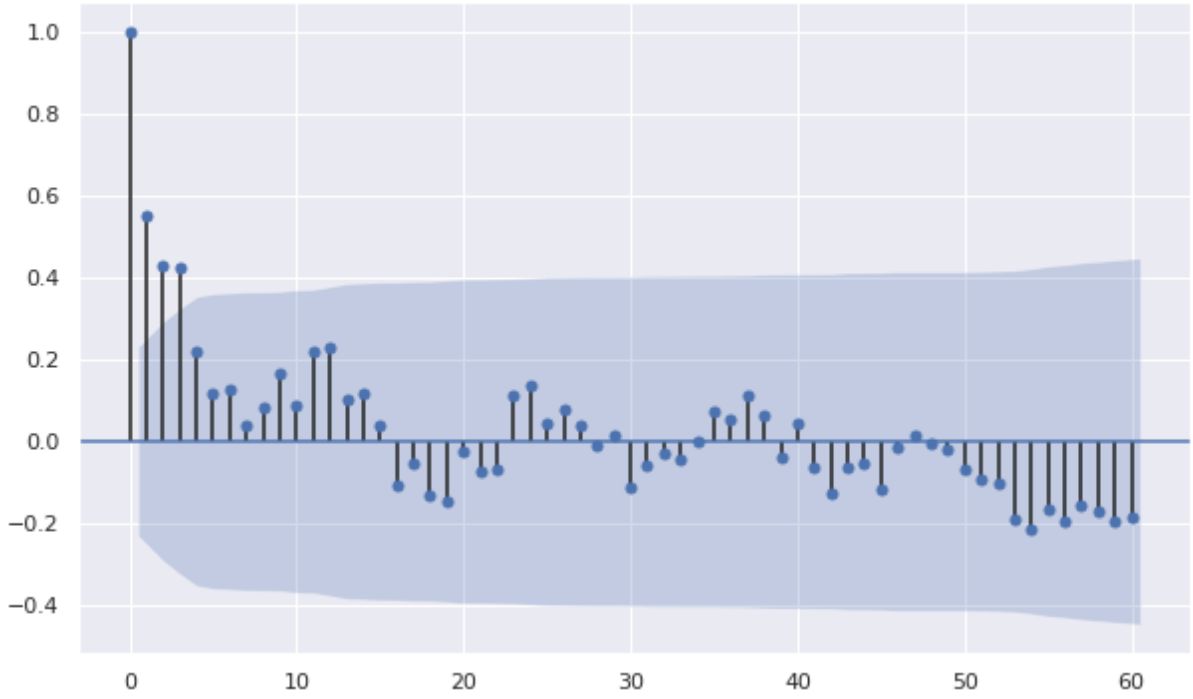
Fonte: Autor

Figura 97 - Decomposição Sazonal Multiplicativa do Perfil de Consumo, Detalhe: Setor N1 = Serviço Público e UF = RS



Fonte: Autor

Figura 98 – Autocorrelação, Detalhe: Setor N1 = Serviço Público e UF = RS



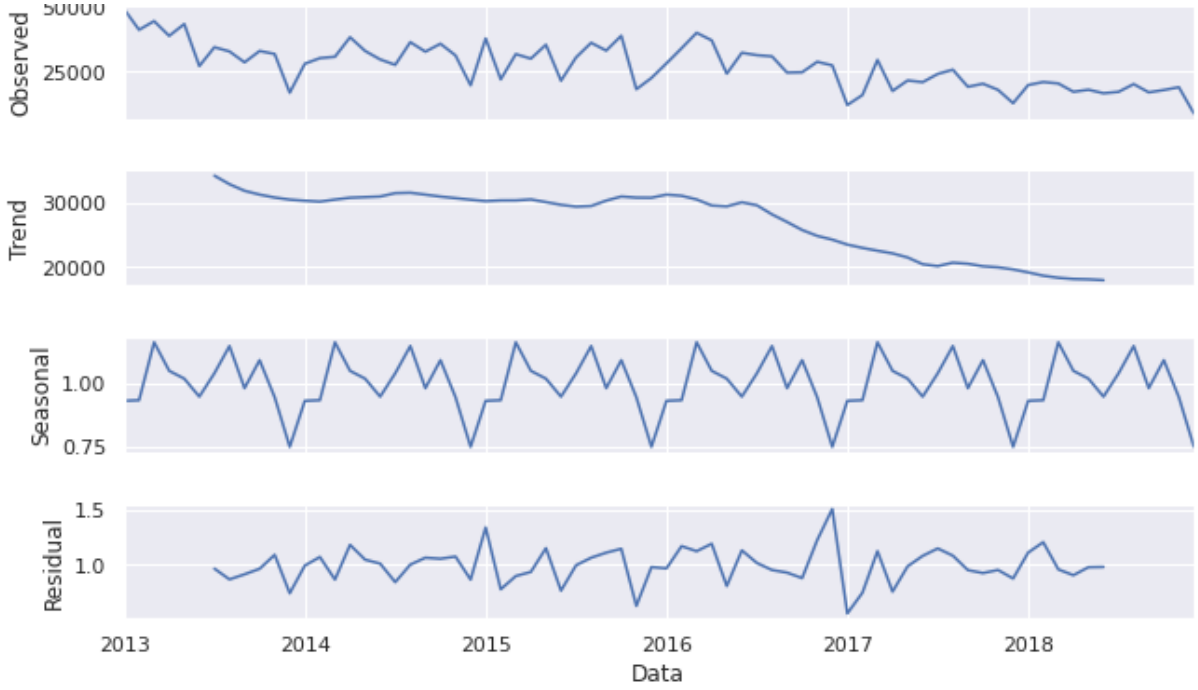
Fonte: Autor

Consumo Próprio

Figura 99 - Perfil de Consumo Mensal, Detalhe: Setor N1 = Consumo Próprio, UF = RS

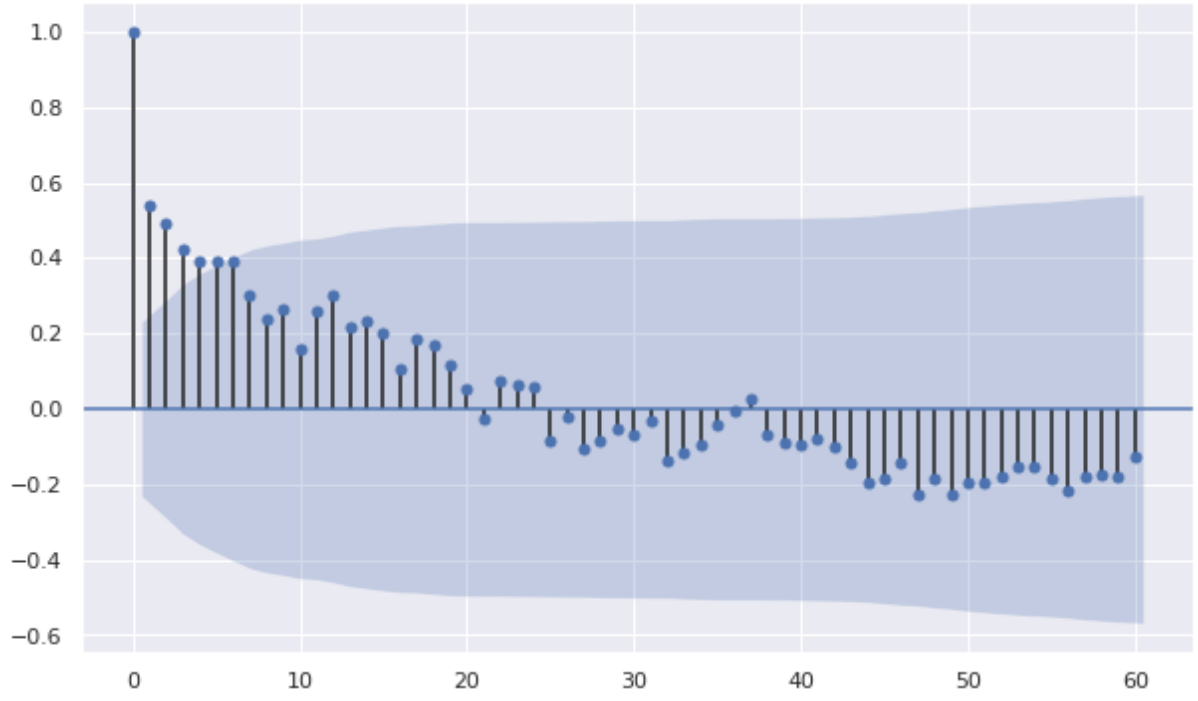
Fonte: Autor

Figura 100 - Decomposição Sazonal Multiplicativa do Perfil de Consumo, Detalhe: Setor N1 = Consumo Próprio e UF = RS



Fonte: Autor

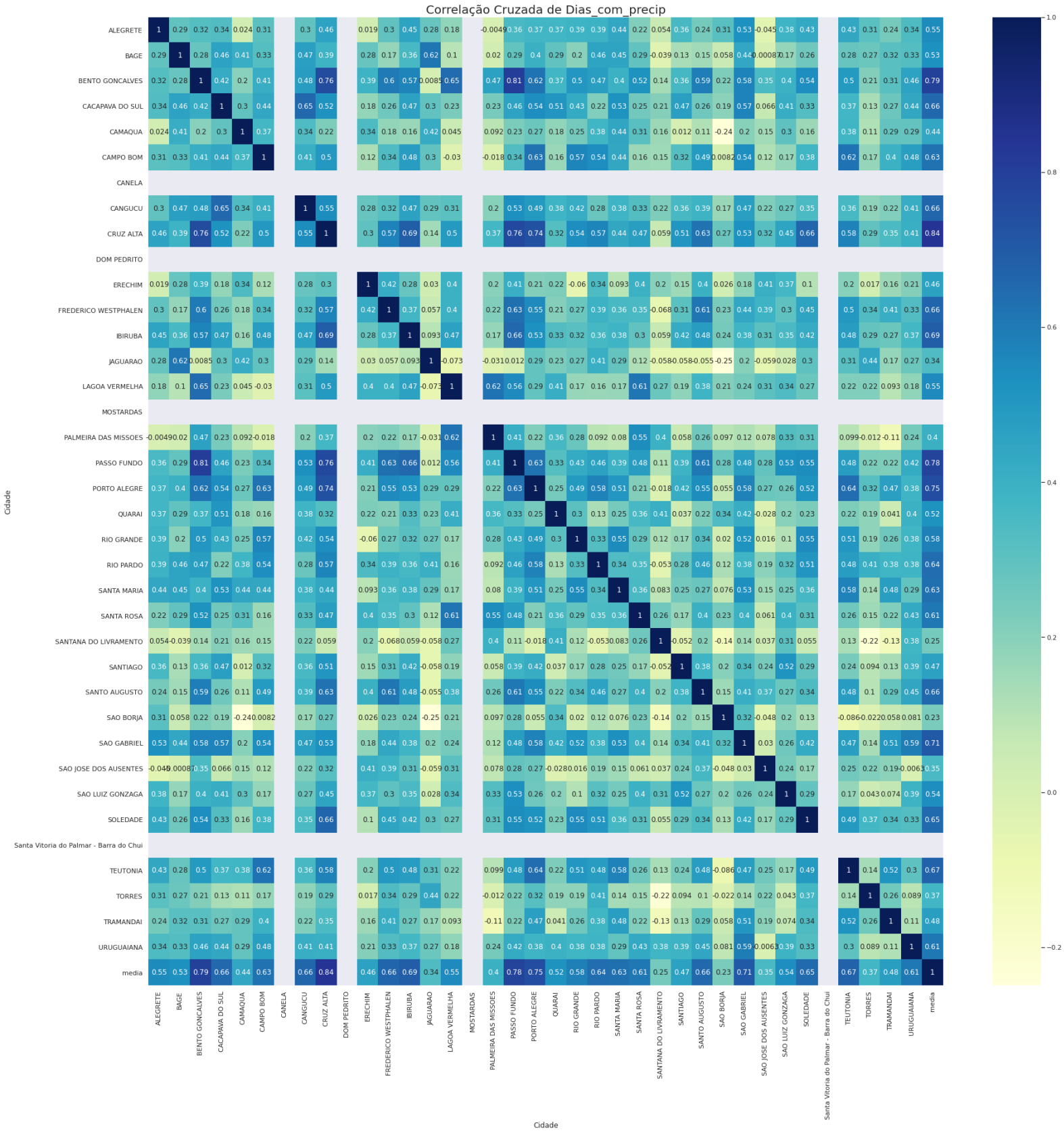
Figura 101 – Autocorrelação, Detalhe: Setor N1 = Consumo Próprio e UF = RS



Fonte: Autor

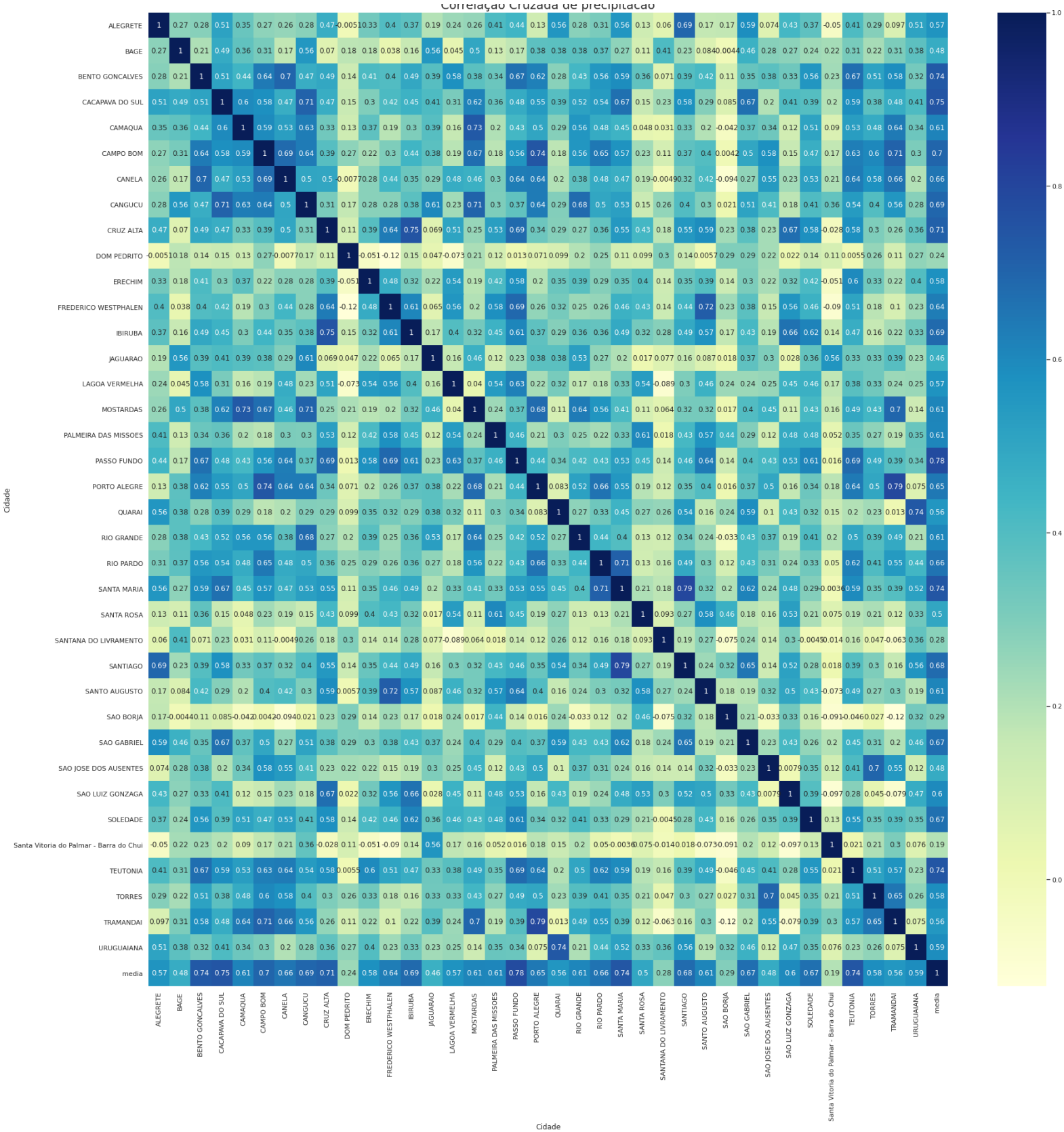
Anexo 2 – Mapas de Correlação Meteorológicos

Figura 102 - Mapa de Correlação Cruzada do Total de Dias com Precipitação por Mês no RS de 2013 a 2018 por Estação Meteorológica.



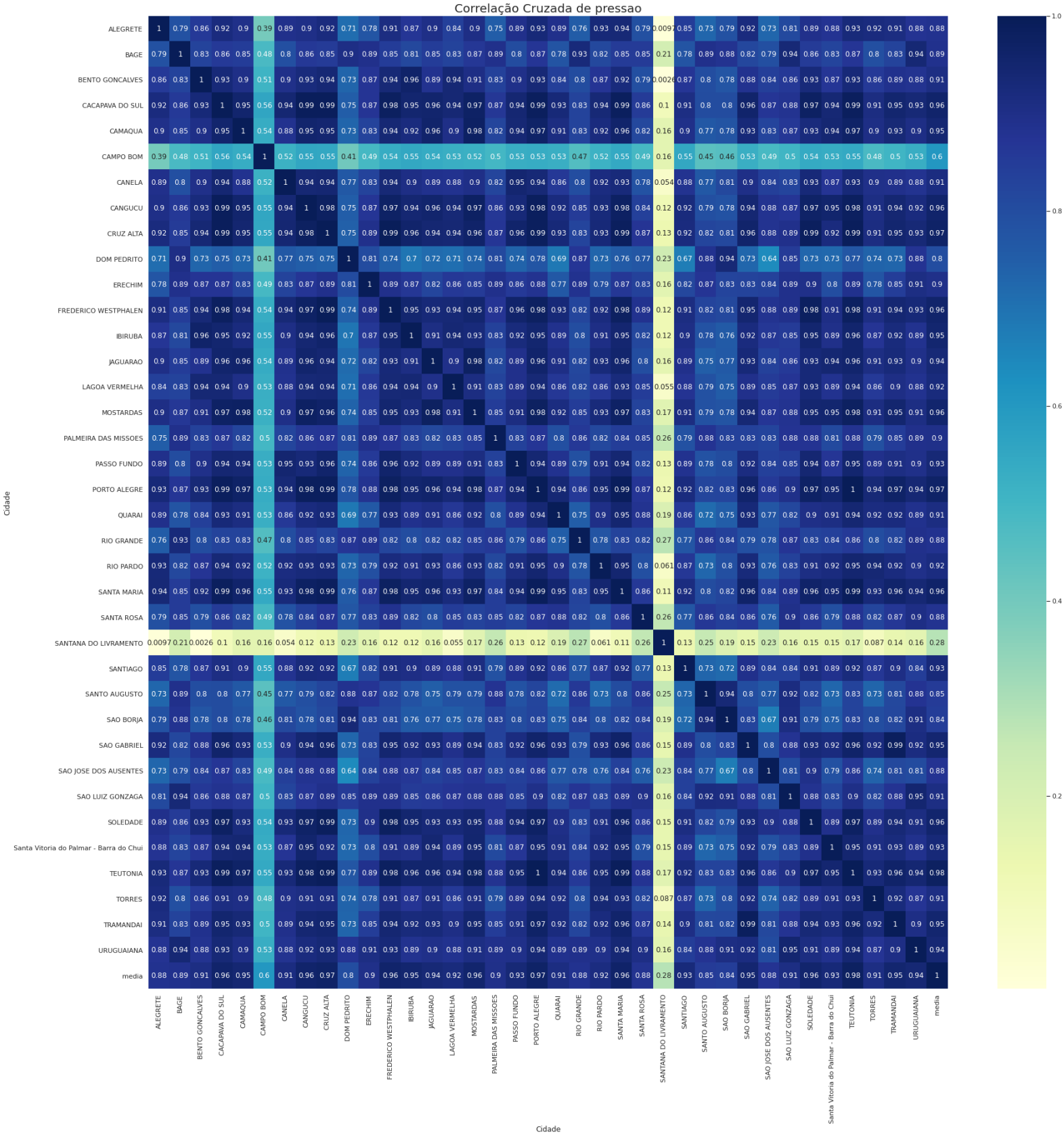
Fonte: Autor

Figura 103 - Mapa de Correlação Cruzada do Acumulado de Precipitação por Mês no RS de 2013 a 2018 por Estação Meteorológica.



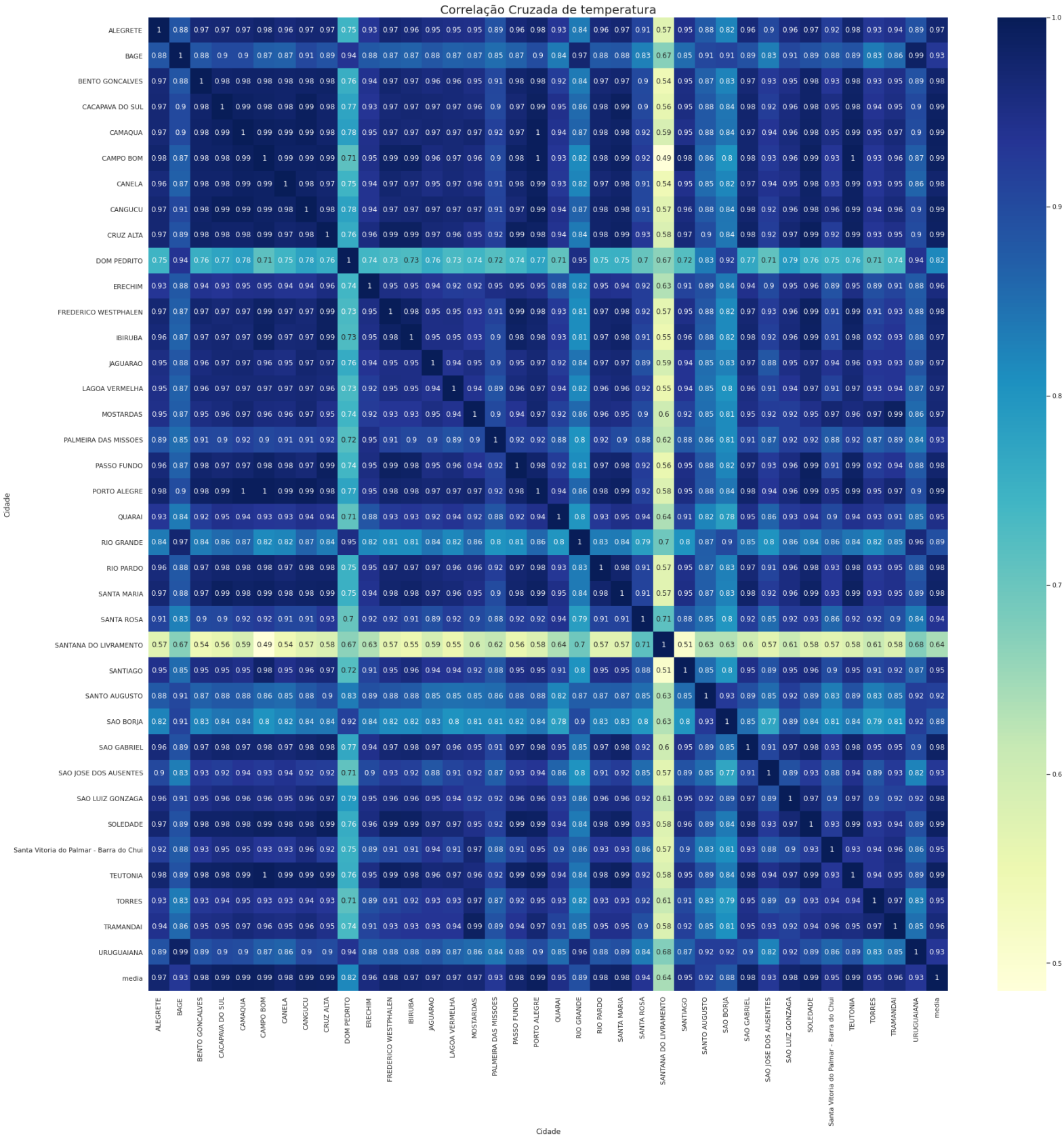
Fonte: Autor

Figura 104 - Mapa de Correlação Cruzada da Média de Pressão por Mês no RS de 2013 a 2018 por Estação Meteorológica.



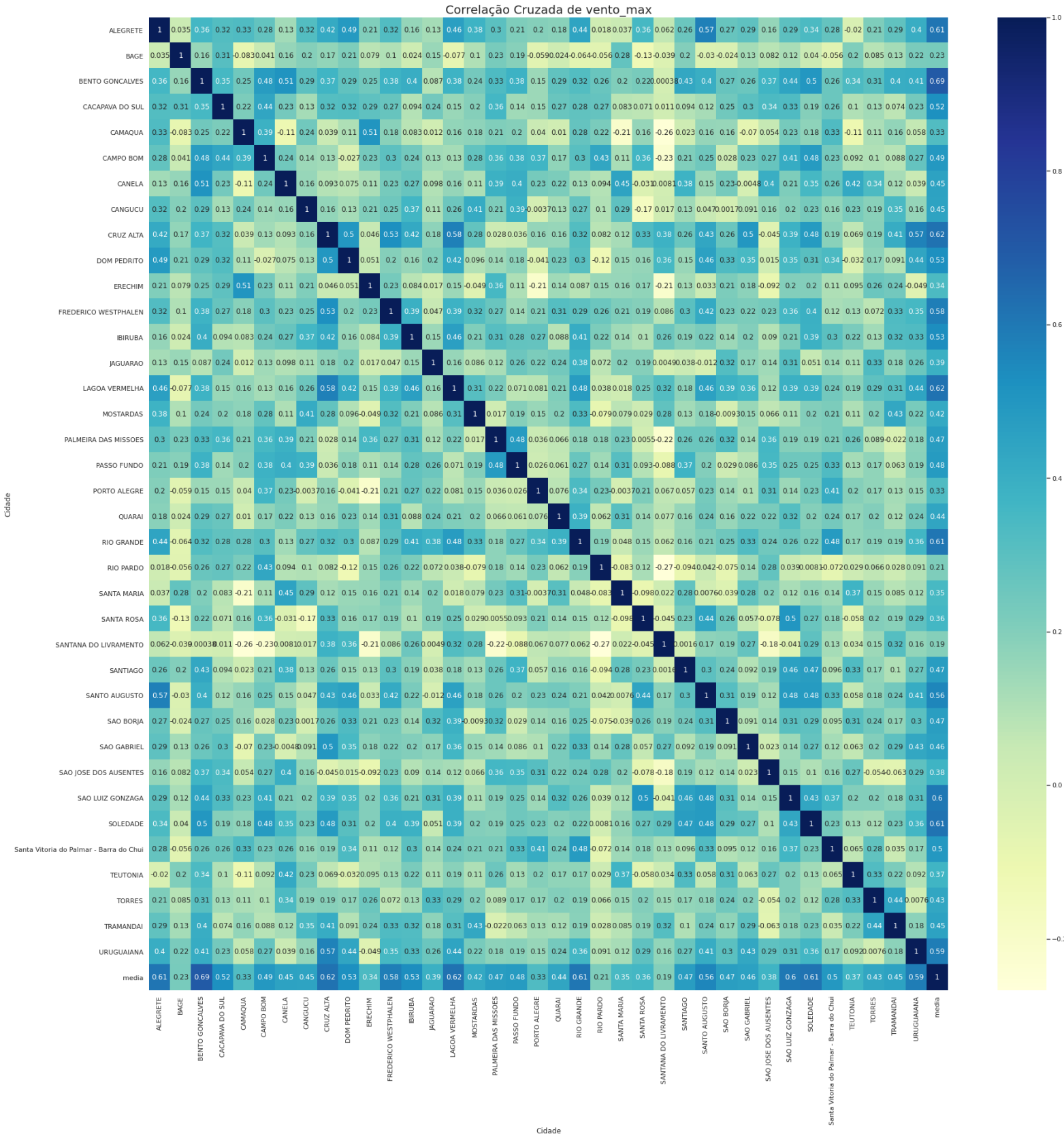
Fonte: Autor

Figura 105 - Mapa de Correlação Cruzada da Média de Temperatura por Mês no RS de 2013 a 2018 por Estação Meteorológica.



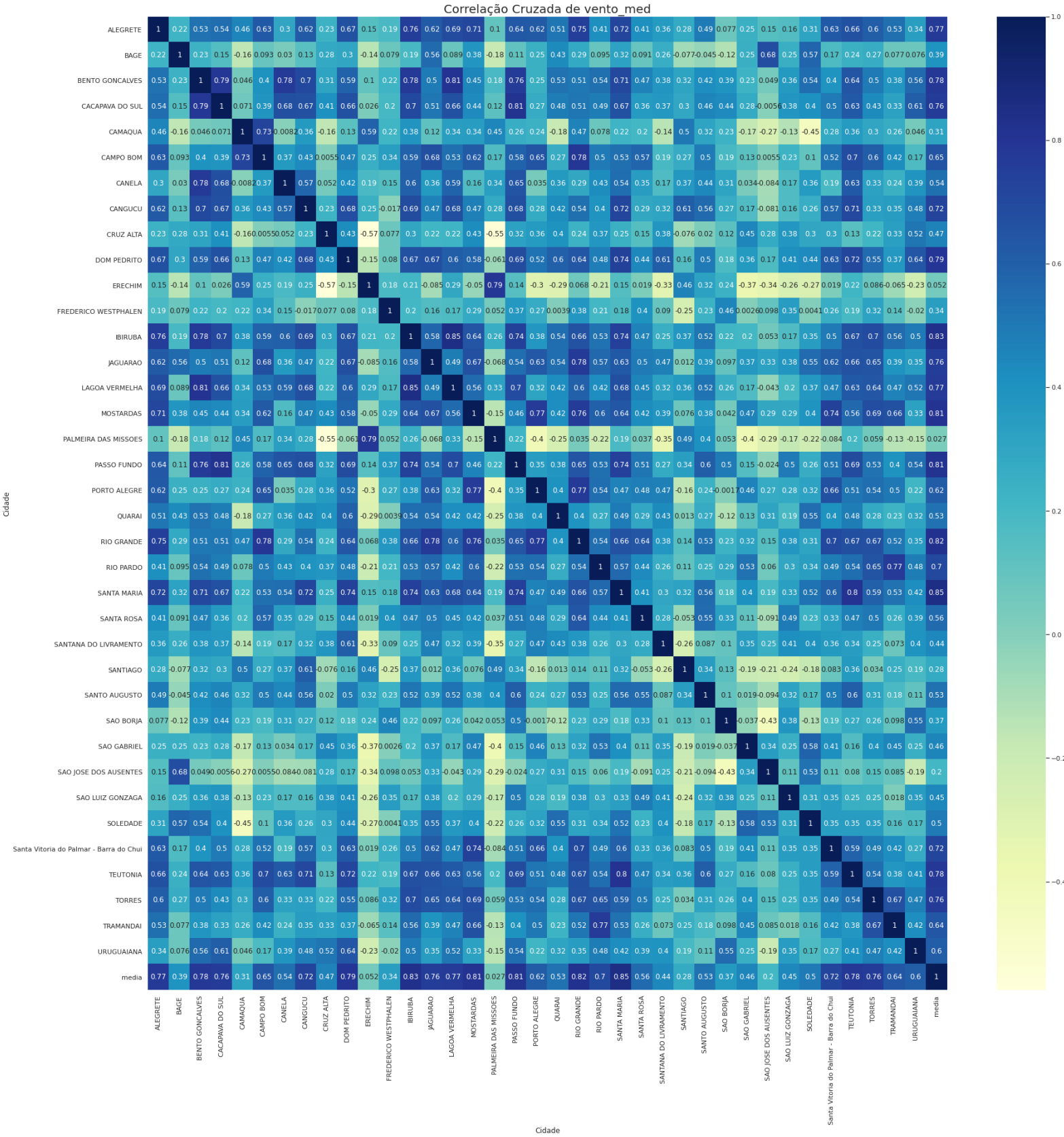
Fonte: Autor

Figura 106 - Mapa de Correlação Cruzada da Velocidade Máxima do Vento por Mês no RS de 2013 a 2018 por Estação Meteorológica.



Fonte: Autor

Figura 107 - Mapa de Correlação Cruzada da Velocidade Média do Vento por Mês no RS de 2013 a 2018 por Estação Meteorológica.

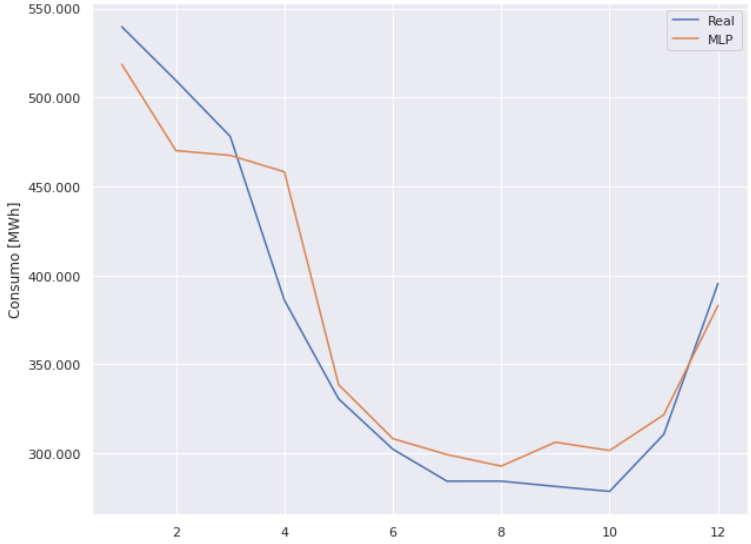


Fonte: Autor

Anexo 3 – Resultado Teste: 4 camandas MLP caso: RS: Rural

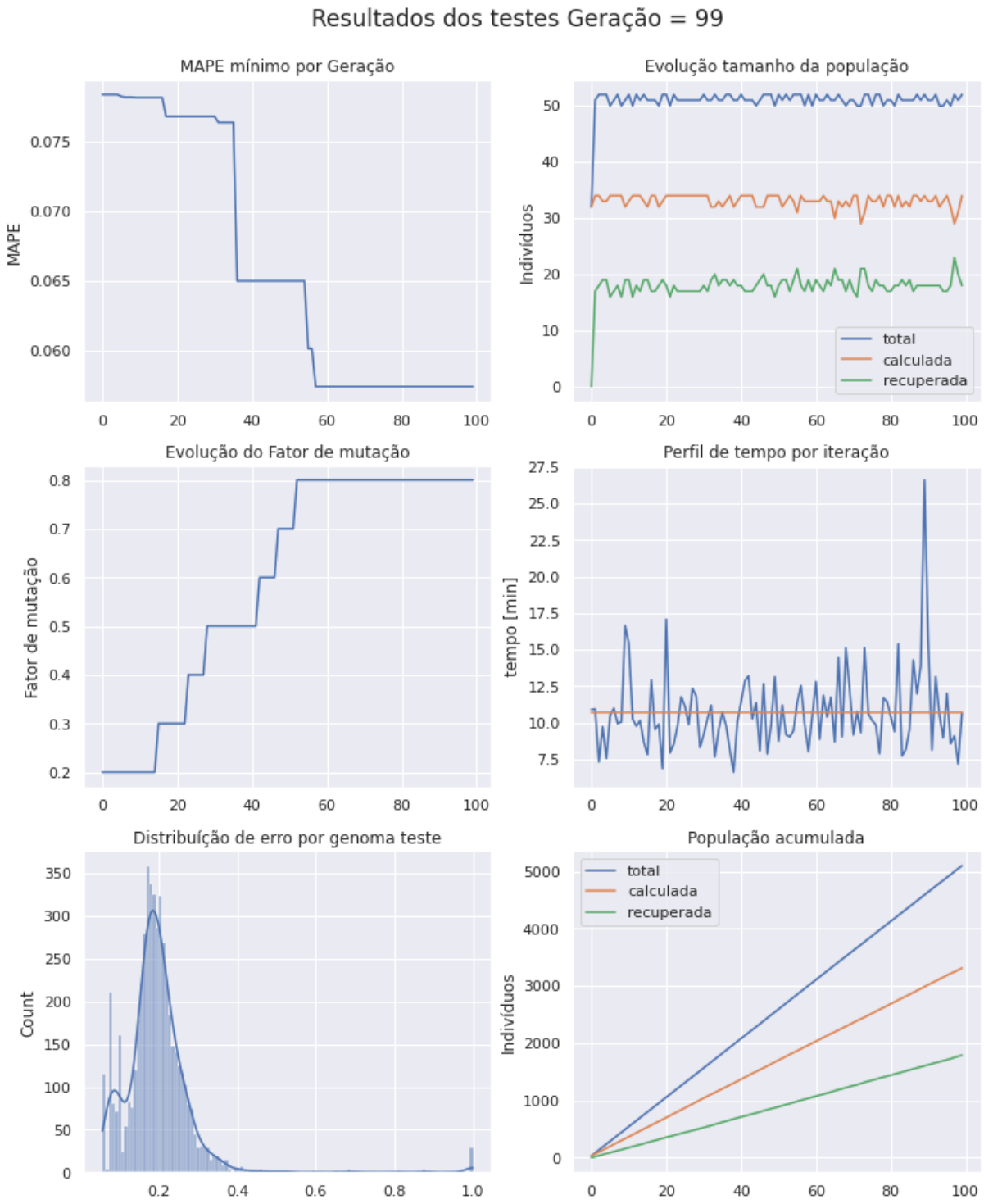
```
->Algoritmo genético de otimização ----------------------------------
     Nova População                          =  True
     Recupera memória de iterações passadas  =  True
     Gerações máximas                        =  100
     Mais aptos selecionados em memória      =  0
     Tamanho da população                    =  32
     Número de Genes                         =  4
     Limite de valores para cada Gene        =  (0, 64)
     Probabilidade de Mutação inicial        =  0.2
     Casais por época                        =  16
     Limite de épocas parado no mesmo erro   =  100
     Erro Alvo                               =  1e-05
-----------------------------------------------------------------------


Menor MAPE   0.057478

CPU times: time out

Wall time: 24h 00min 00s
```

Figura 108 - Teste Valor Real vs Predito MLP



Fonte: Autor

Figura 109 - Resultado da Otimização por Algoritmo Genético



Fonte: Autor

Anexo 4 – Resultado Teste: 2 camadas (Bi) LSTM e 4 camadas MLP com *dropout* caso:

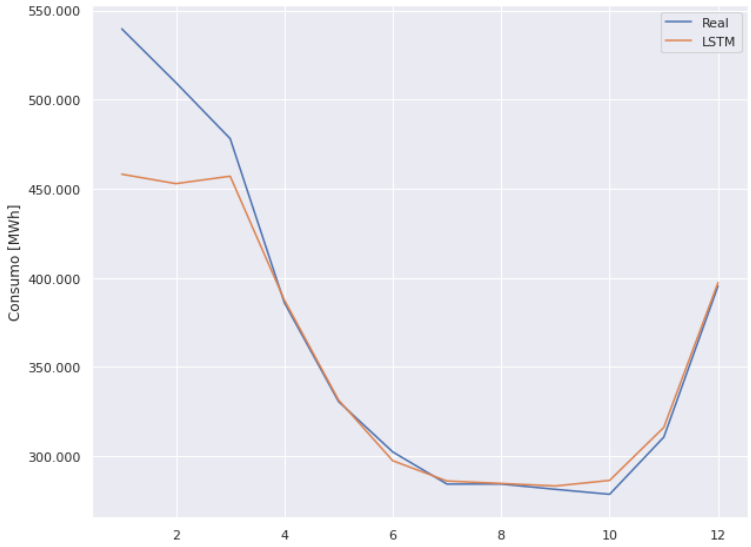
RS: Rural

```
->Algoritmo genético de otimização ----------------------------------
     Nova População                          =  True
     Recupera memória de iterações passadas  =  False
     Gerações máximas                        =  200
     Mais aptos selecionados em memória      =  0
     Tamanho da população                    =  32
     Número de Genes                         =  13
     Limite de valores para cada Gene        =  (0, 64)
     Probabilidade de Mutação inicial        =  0.2
     Casais por época                        =  8
     Limite de épocas parado no mesmo erro   =  200
     Erro Alvo                               =  1e-05
----------------------------------------------------------------------
```

Estrutura:

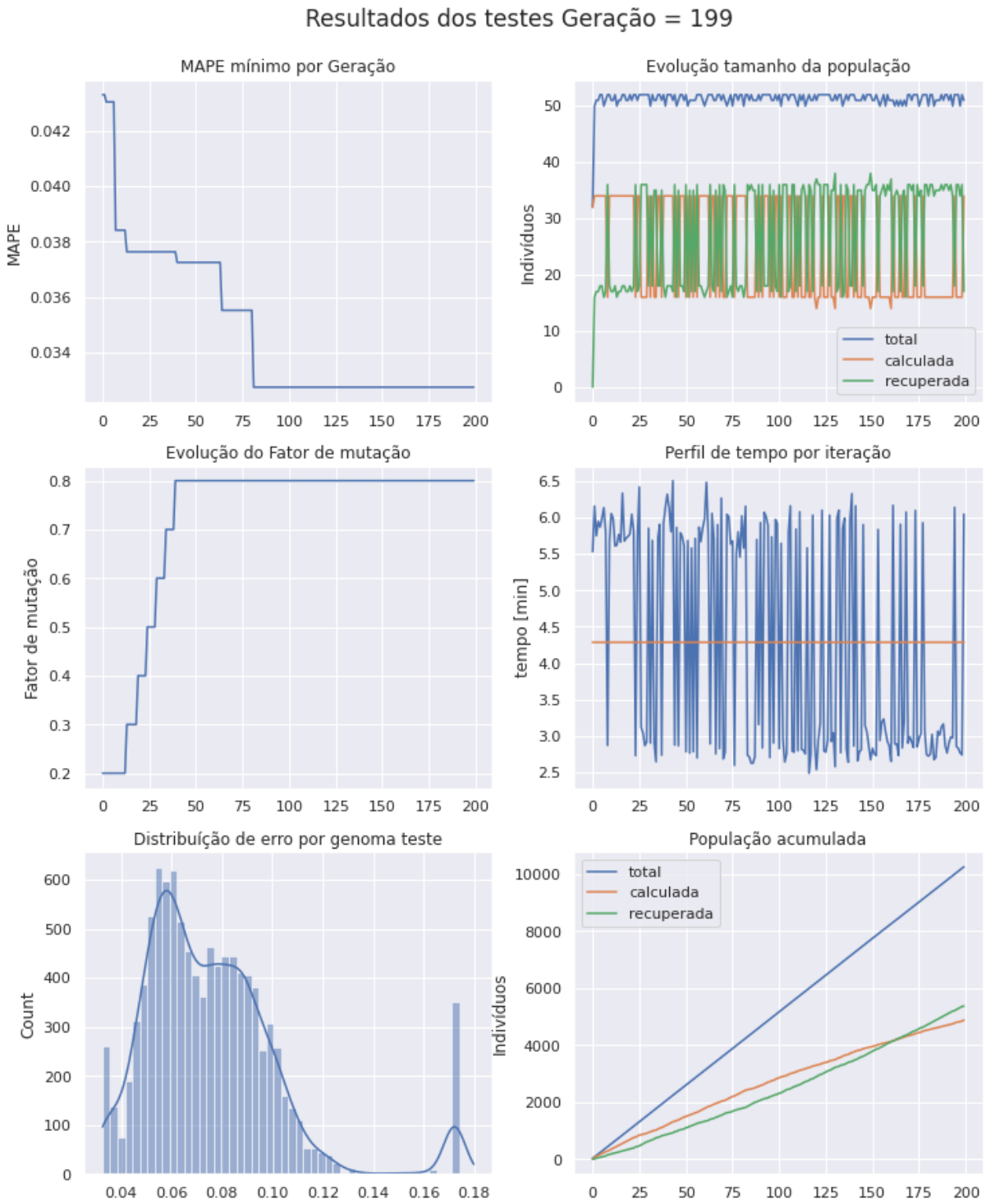
```
->BILSTM(41)->d(0.1)->LSTM(49)->d(0.1)->MLP(44)->d(0.2)->MLP(46)->d(0.1)-
>MLP(41)->d(0.1)->MLP(28)->

Menor MAPE   0.032756552045235796

CPU times: user 16h 25min 41s, sys: 43min 37s, total: 17h 9min 19s

Wall time: 14h 27min 39s
```

Figura 110 - Teste Valor Real vs Predito LSTM



Fonte: Autor

Figura 111 - Resultado da Otimização por Algoritmo Genético



Fonte: Autor

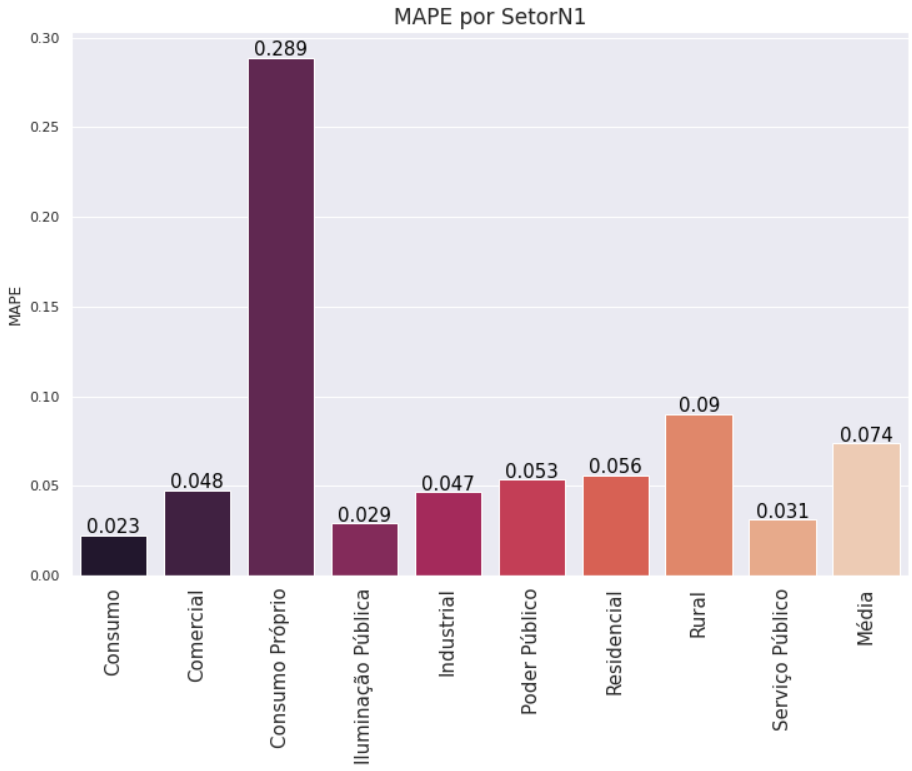
Anexo 5 – Resultado Teste: 2 camadas (Bi) LSTM e 4 camadas MLP com *Dropout* caso: RS Estratificado Com Setores N1, Rede Única para Todos os Setores

```
->Algoritmo genético de otimização ----------------------------------
       Nova Populaçao                          =  True
       Recupera memória de iterações passadas  =  False
       Gerações máximas                        =  100
       Mais aptos selecionados em memória      =  0
       Tamanho da população                    =  64
       Número de Genes                         =  13
       Limite de valores para cada Gene        =  (0, 64)
       Probabilidade de Mutação inicial        =  0.2
       Casais por época                        =  16
       Limite de épocas parado no mesmo erro   =  200
       Erro Alvo                               =  1e-05
-------------------------------------------------------------------------
```

Estrutura:

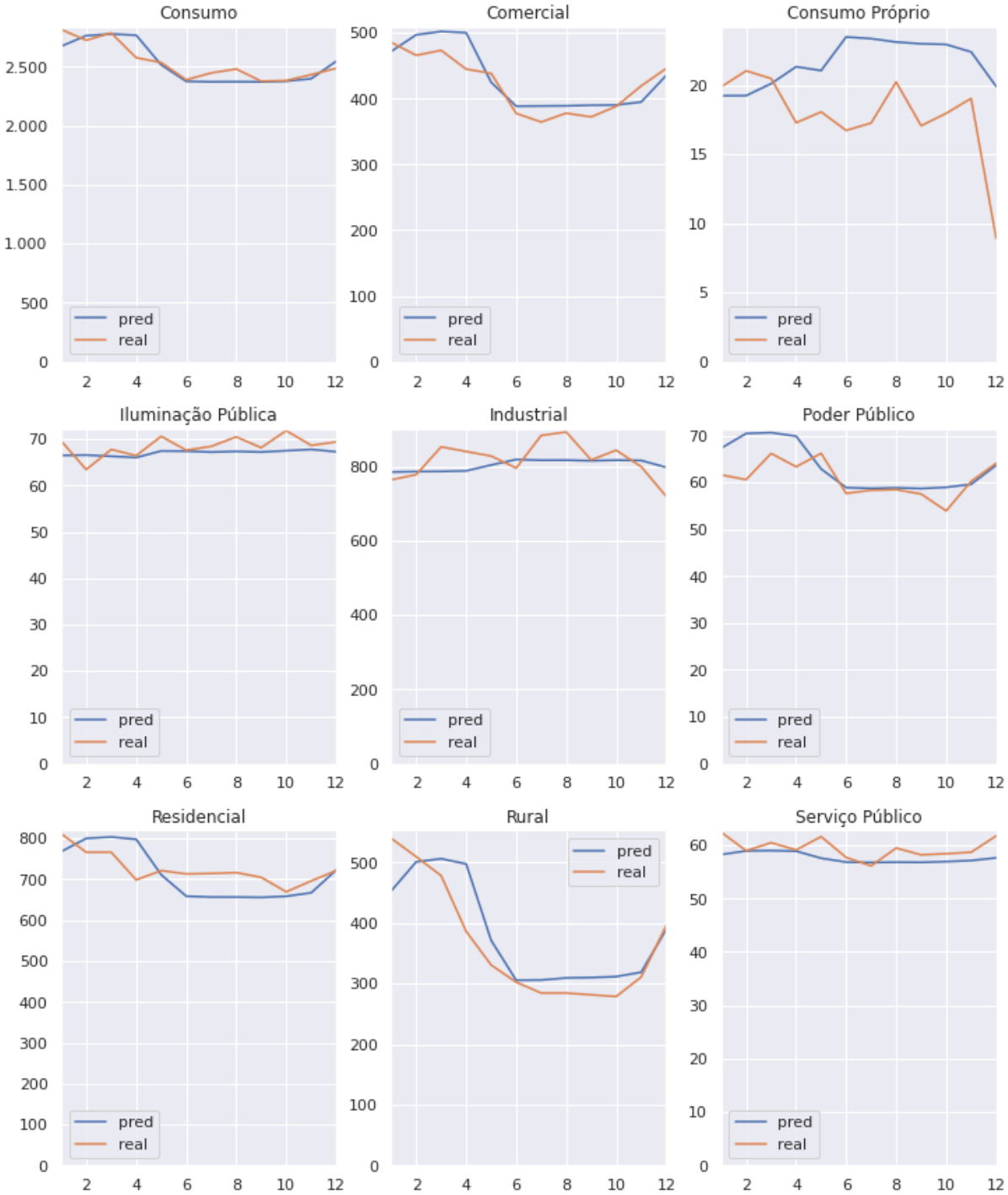
```
->LSTM(50)->d(0.1)->LSTM(46)->d(0.2)->MLP(62)->d(0.2)->MLP(50)->d(0.1)->MLP(42)->d(0.1)->MLP(51)->

Menor Mape   0.057823467627559556

Código   [23 47 63 28 16 47  9 50 46 62 50 42 51]

CPU times: user 20h 29min 11s, sys: 54min 3s, total: 21h 23min 15s

Wall time: 17h 22min
```

Figura 112 – Erro por Setor N1



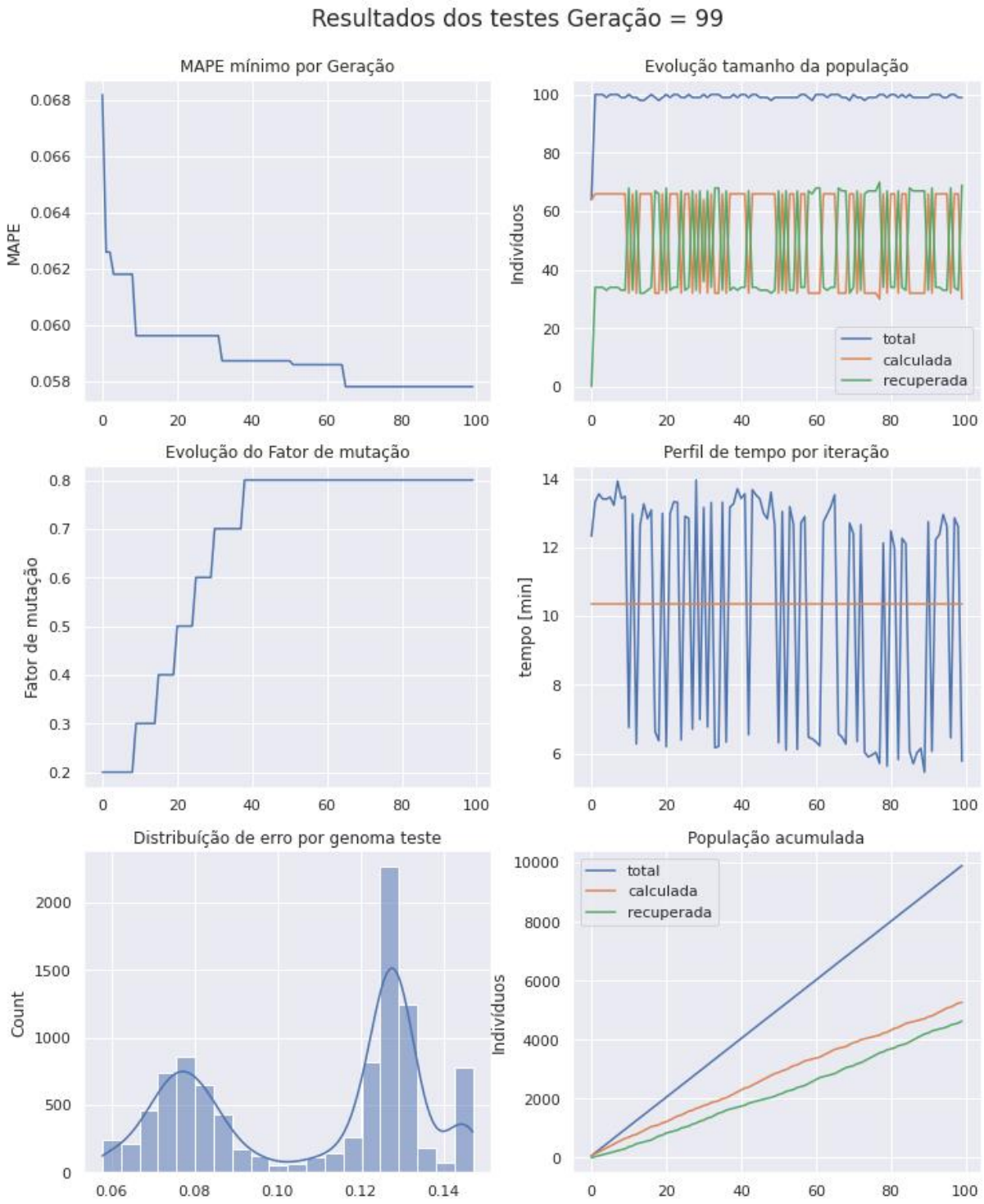
Fonte: Autor

Figura 113 - Teste Valor Real vs Predito LSTM



Fonte: Autor

Figura 114 - Resultado da Otimização por Algoritmo Genético



Fonte: Autor

Anexo 6 – Resultado Teste: 2 camadas (Bi) LSTM e 4 camadas MLP com *Dropout* caso: RS Estratificado Com Setores N1, Uma rede para cada Setor

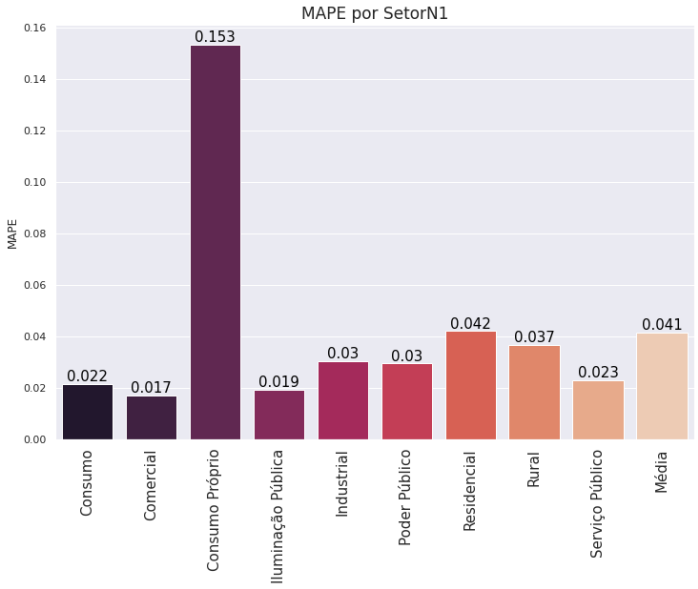
```
->Algoritmo genético de otimização ----------------------------------
       Nova Populaçao                          =  True
       Recupera memória de iterações passadas  =  False
       Gerações máximas                        =  32
       Mais aptos selecionados em memória      =  0
       Tamanho da população                    =  32
       Número de Genes                         =  13
       Limite de valores para cada Gene        =  (0, 64)
       Probabilidade de Mutação inicial        =  0.2
       Casais por época                        =  16
       Limite de épocas parado no mesmo erro   =  200
       Erro Alvo                               =  1e-05
-----------------------------------------------------------------------
Tempo de Processamento: 22h52m


Consumo:
MAPE       =  0.021641
gen        =  [39 58 56 16 63 44 50 26 39 40  2 18  1]
->LSTM(26)->d(0.2)->BILSTM(39)->d(0.2)->MLP(40)->d(0.1)->MLP(2)->d(0.2)-
>MLP(18)->d(0.2)->MLP(1)->

Comercial:
MAPE       =  0.017043
gen        =  [31  0 55 55 39 57  3 12  0 42  0 24 18]
->LSTM(12)->d(0.1)->MLP(42)->d(0.1)->MLP(24)->d(0.1)->MLP(18)->

Consumo Próprio:
MAPE       =  0.153316
gen        =  [60 62 35 56 26 21 47 23  9 61 41 19 34]
->BILSTM(23)->d(0.1)->BILSTM(9)->d(0.2)->MLP(61)->d(0.2)->MLP(41)->d(0.1)-
>MLP(19)->d(0.1)->MLP(34)->

Iluminação Pública:
MAPE       =  0.019208
gen        =  [54 54 55 20 26 54 25  4 61 47 50 49 11]
->BILSTM(4)->d(0.1)->BILSTM(61)->d(0.2)->MLP(47)->d(0.2)->MLP(50)->d(0.2)-
>MLP(49)->d(0.1)->MLP(11)->

Industrial:
MAPE       =  0.03043
gen        =  [37 42 23 54 23  1  7  0 58 48 23 50 39]
->BILSTM(58)->d(0.2)->MLP(48)->d(0.1)->MLP(23)->d(0.1)->MLP(50)->d(0.1)-
>MLP(39)->

Poder Público:
MAPE       =  0.029517
gen        =  [39 39 24 22  4 50 54 33 59  8  2  2 27]
->LSTM(33)->d(0.2)->LSTM(59)->d(0.2)->MLP(8)->d(0.2)->MLP(2)->d(0.2)-
>MLP(2)->d(0.2)->MLP(27)->

Residencial:
MAPE       =  0.042109
gen        =  [41 28 62 52  7 50 42 18 60  7  2 20 33]
->LSTM(18)->d(0.2)->BILSTM(60)->d(0.2)->MLP(7)->d(0.1)->MLP(2)->d(0.2)-
>MLP(20)->d(0.2)->MLP(33)->

Rural:
MAPE       =  0.036827
gen        =  [18 28 47 39 58 50  1 21 56 62 42 62 25]
```

```
->BILSTM(21)->d(0.1)->BILSTM(56)->d(0.1)->MLP(62)->d(0.2)->MLP(42)->d(0.2)-
>MLP(62)->d(0.1)->MLP(25)->
```

Serviço Público:

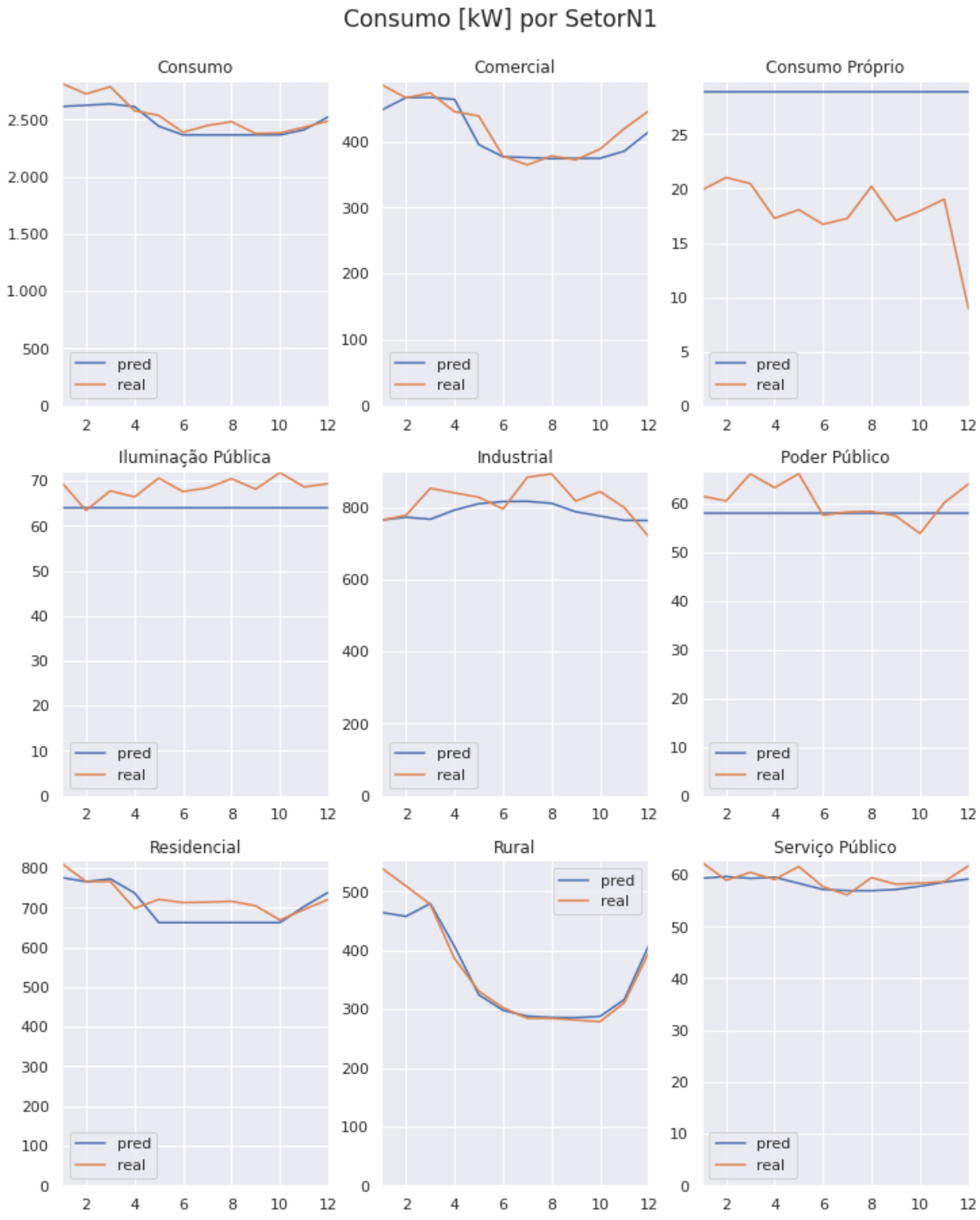
```
MAPE      =  0.023087
id_arca   =  591
gen       =  [22 63 58 13 53 17  0 35 51 63 34 50 26]
->BILSTM(35)->d(0.2)->LSTM(51)->d(0.1)->MLP(63)->d(0.1)->MLP(34)->d(0.1)-
>MLP(50)->d(0)->MLP(26)->

Tempo total de execução: 24h 04min
*estimado.o precesso teve de ser realizado em 2 partes.
```

Figura 115 – Erro por Setor N1



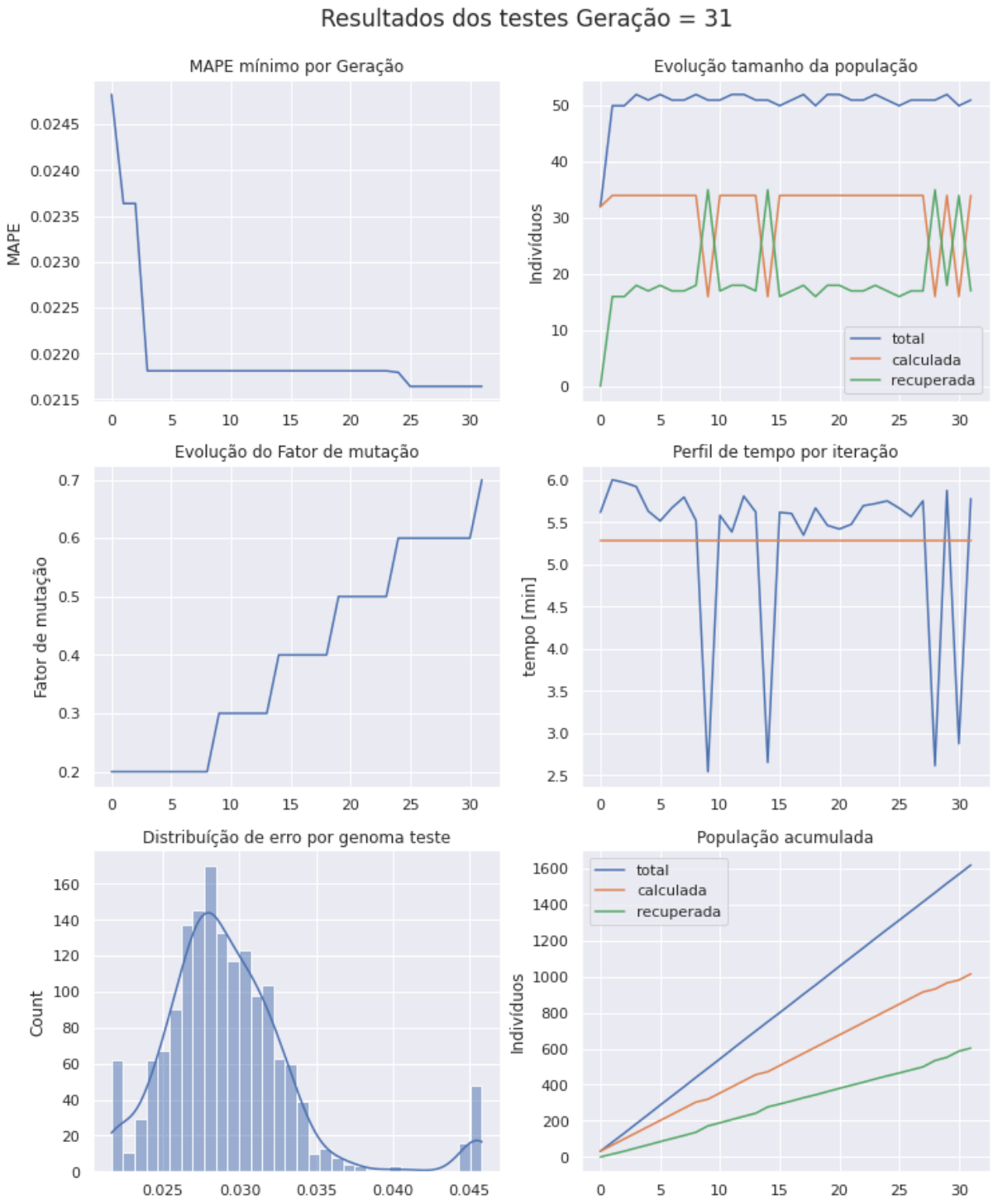
Fonte: Autor

Figura 116 - Teste Valor Real vs Predito LSTM



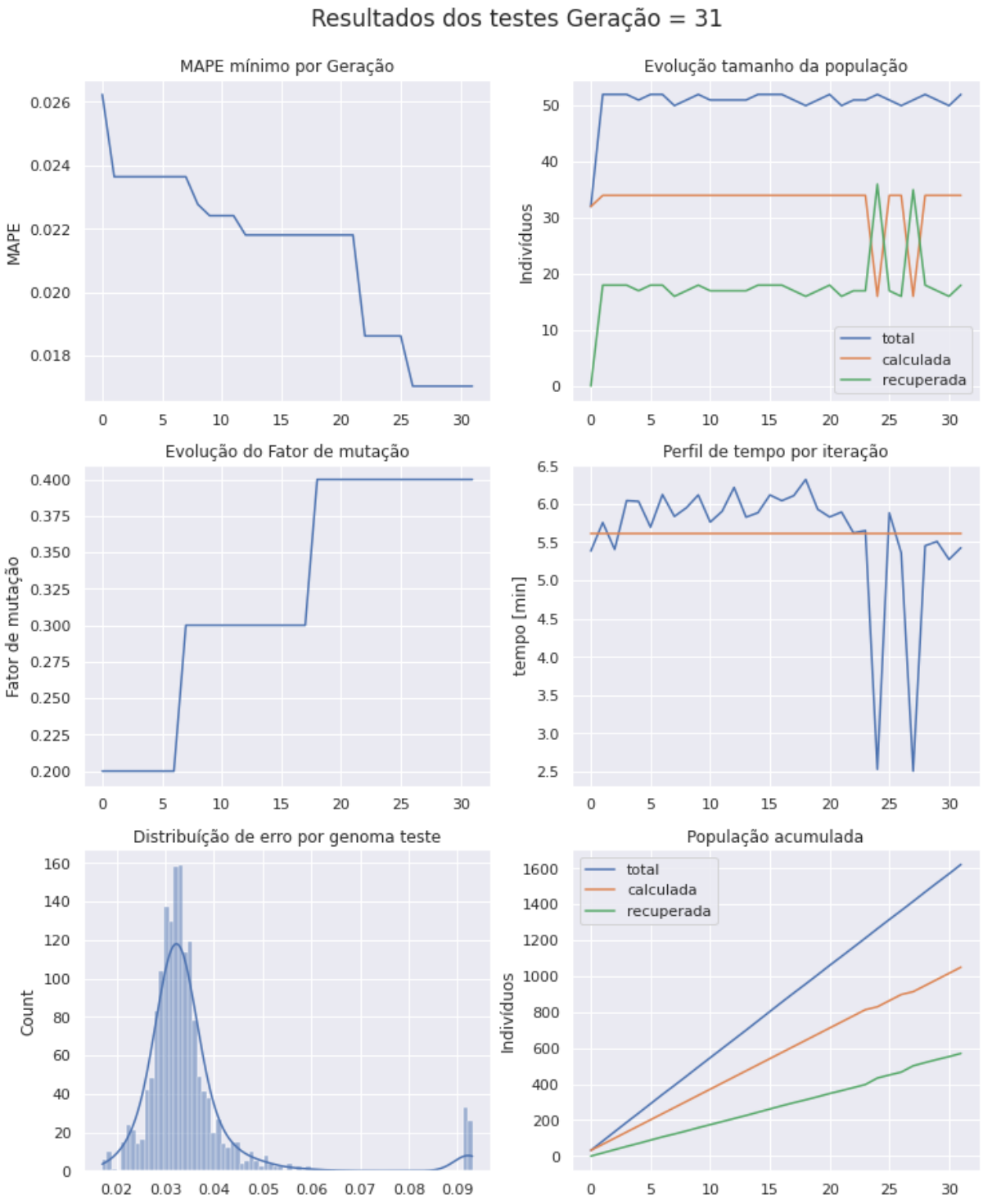
Fonte: Autor

Figura 117 - Resultado da Otimização por Algoritmo Genético (Consumo)



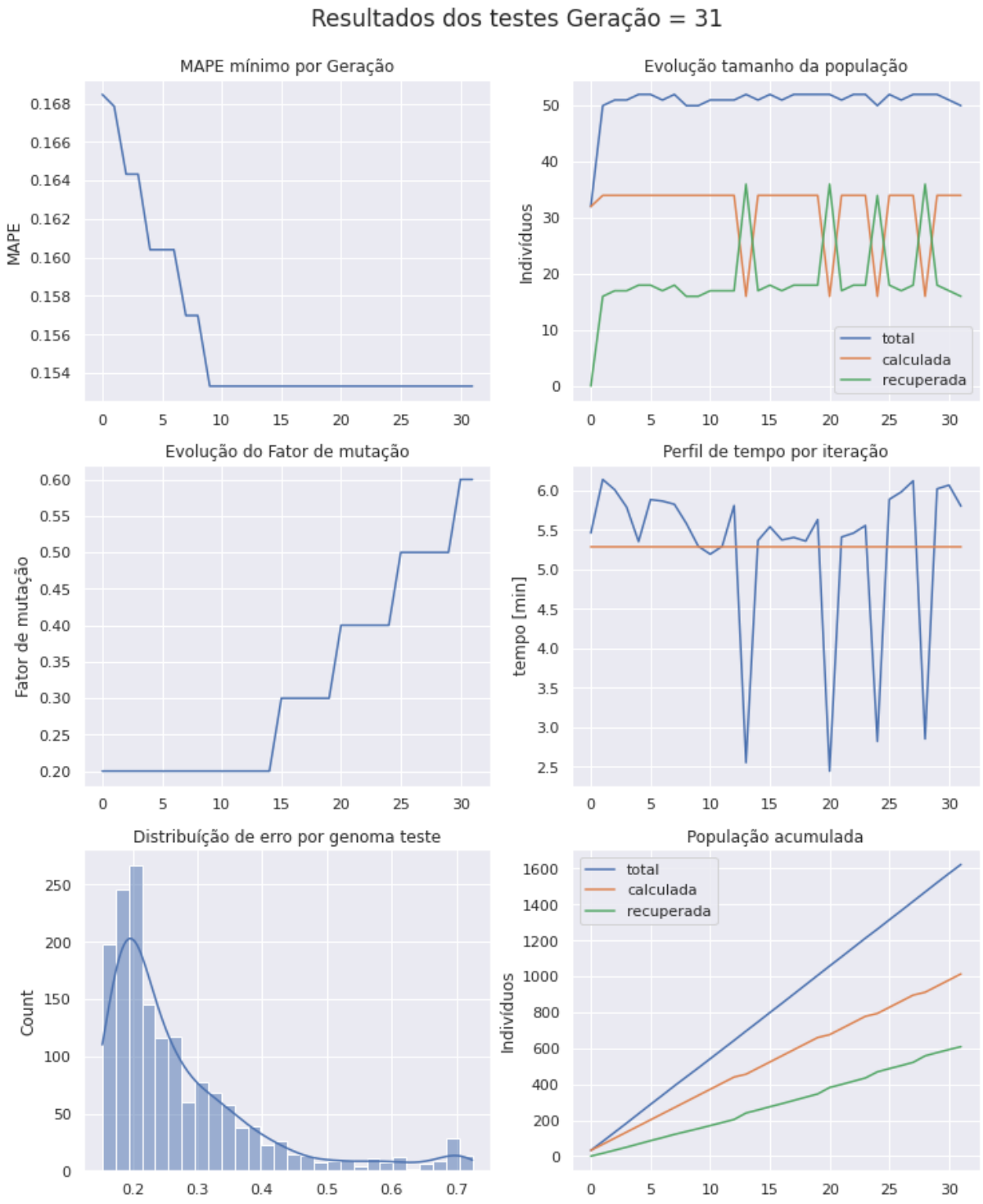
Fonte: Autor

Figura 118 - Resultado da Otimização por Algoritmo Genético (Comercial)



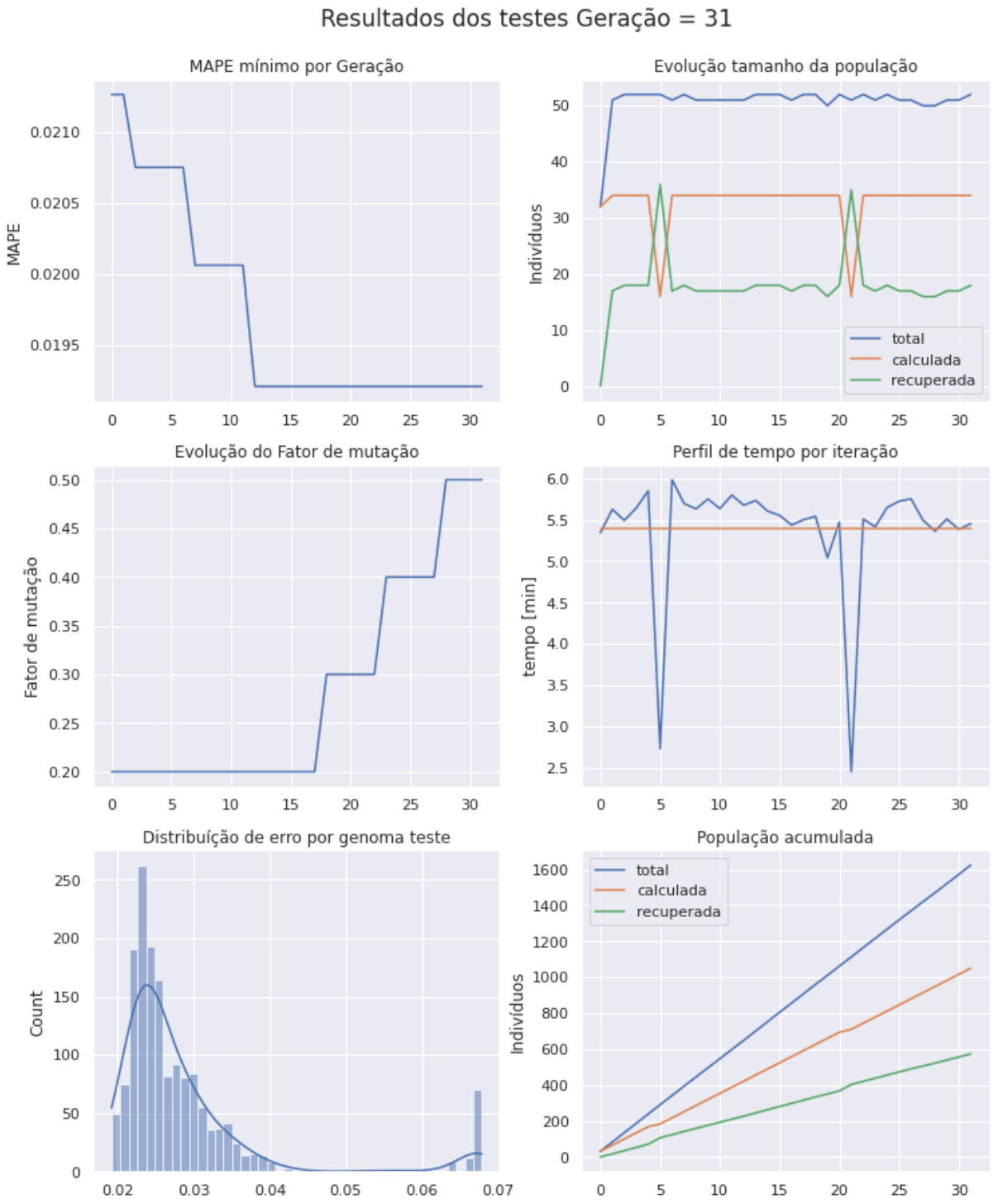
Fonte: Autor

Figura 119 - Resultado da Otimização por Algoritmo Genético (Consumo Próprio)



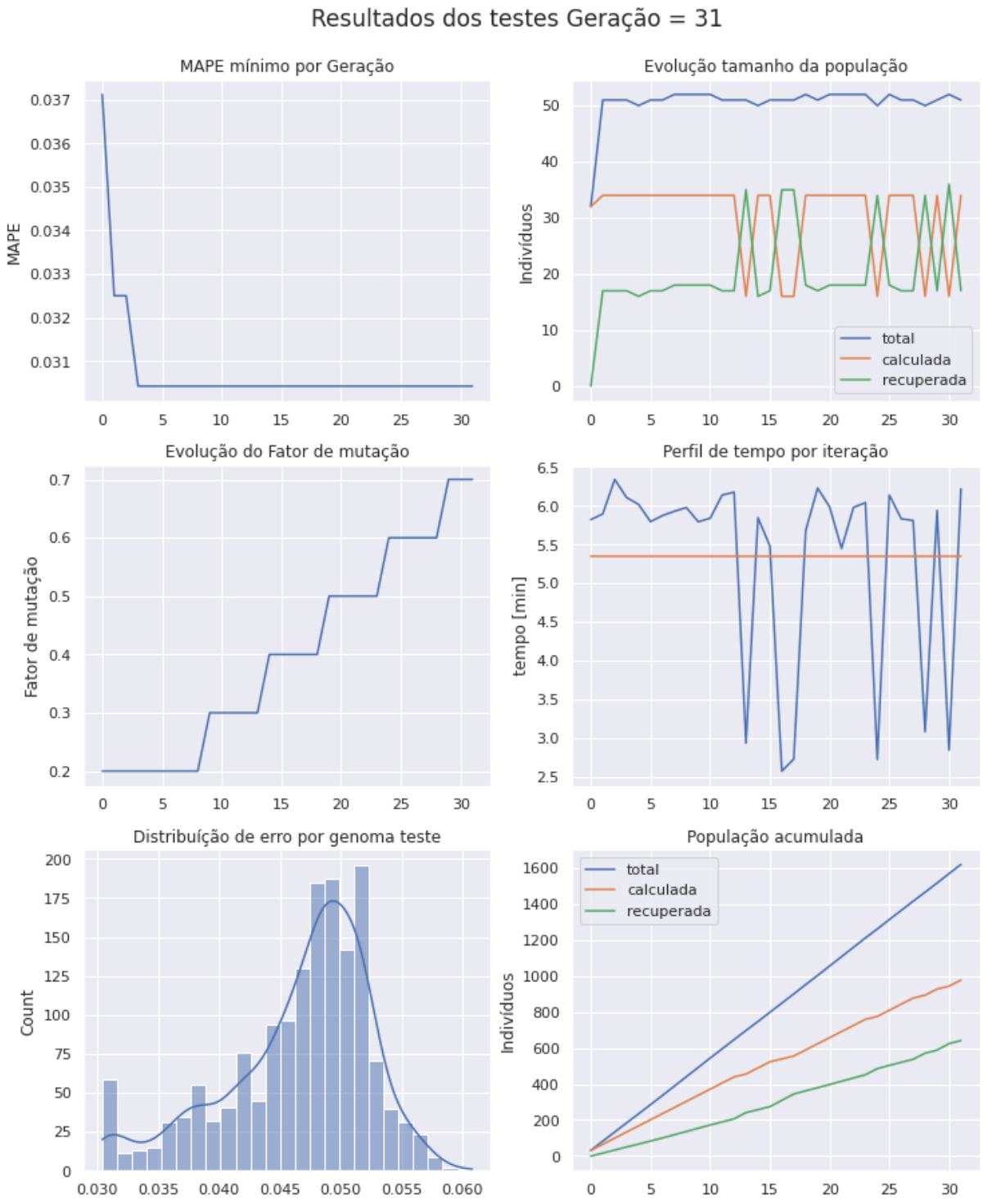
Fonte: Autor

Figura 120 - Resultado da Otimização por Algoritmo Genético (Iluminação Pública)



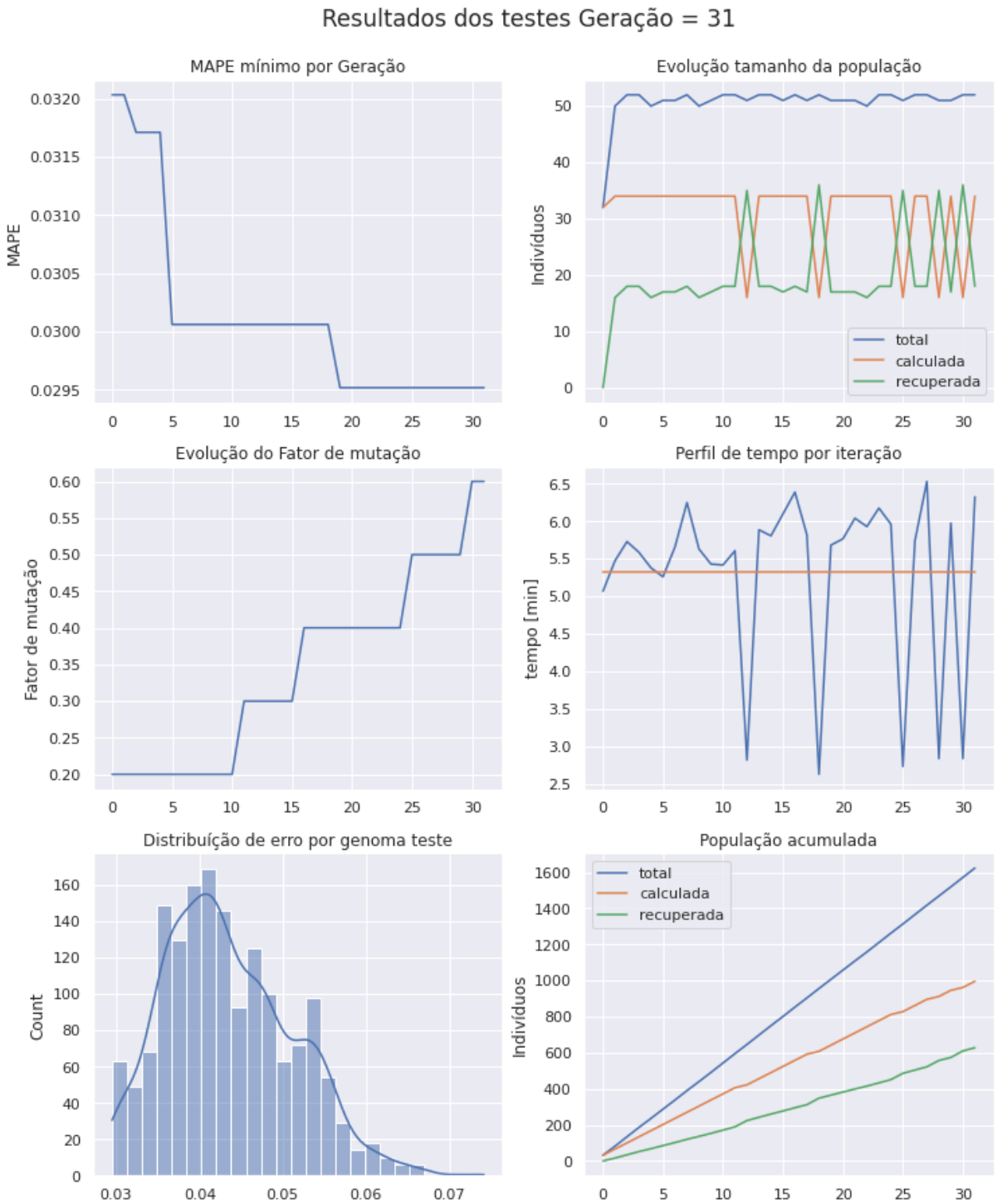
Fonte: Autor

Figura 121 - Resultado da Otimização por Algoritmo Genético (Industrial)



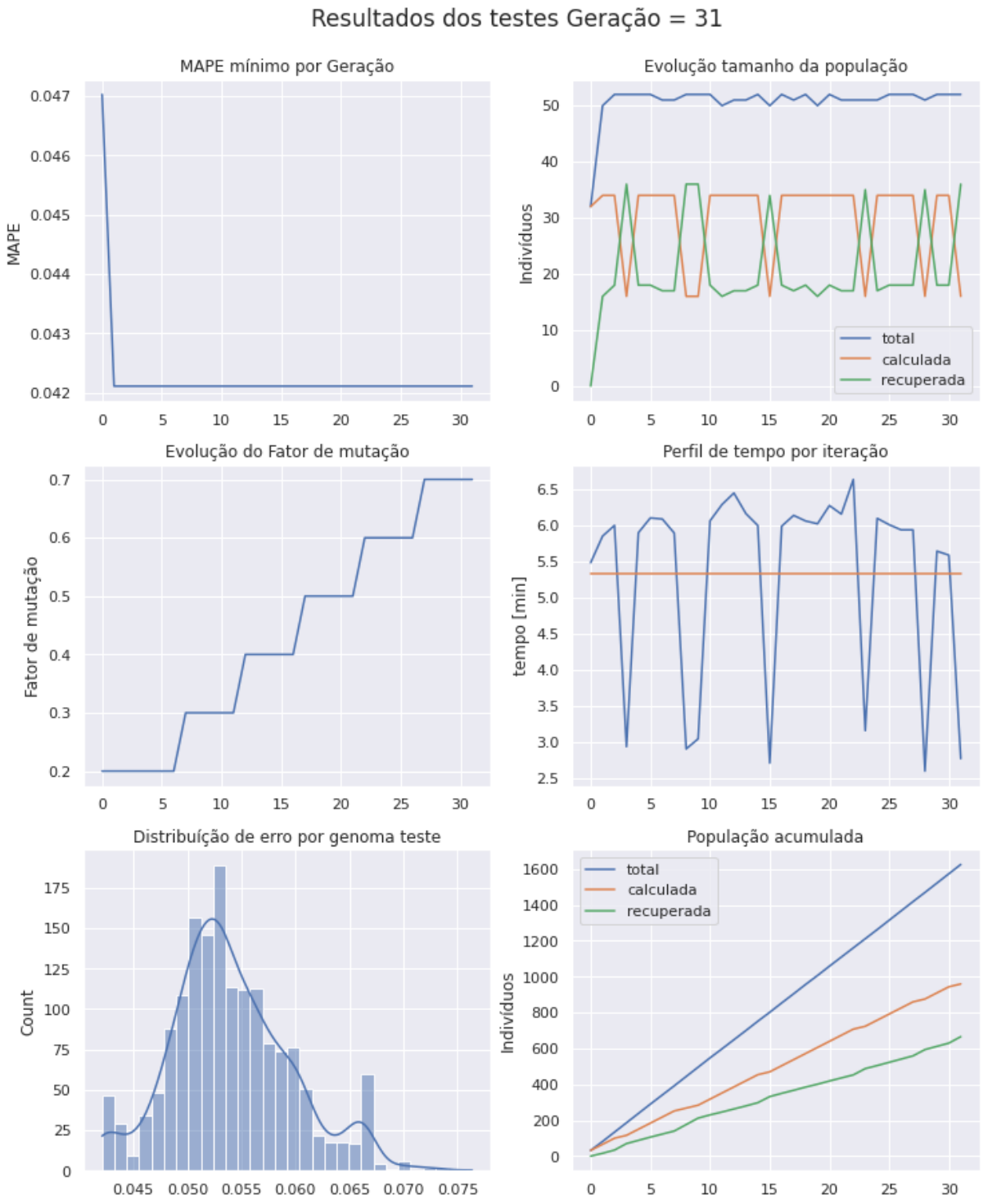
Fonte: Autor

Figura 122 - Resultado da Otimização por Algoritmo Genético (Poder Público)



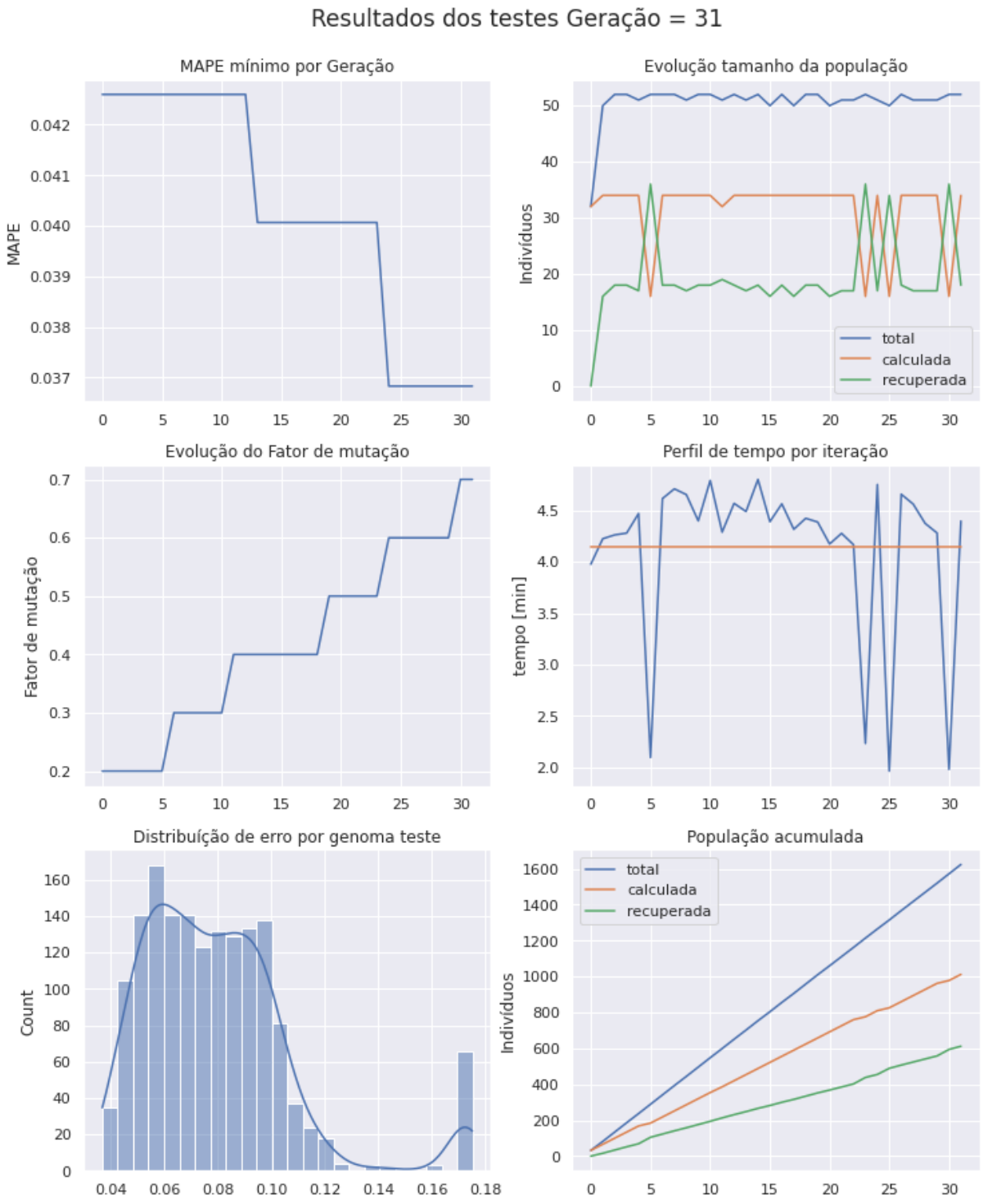
Fonte: Autor

Figura 123 - Resultado da Otimização por Algoritmo Genético (Residencial)



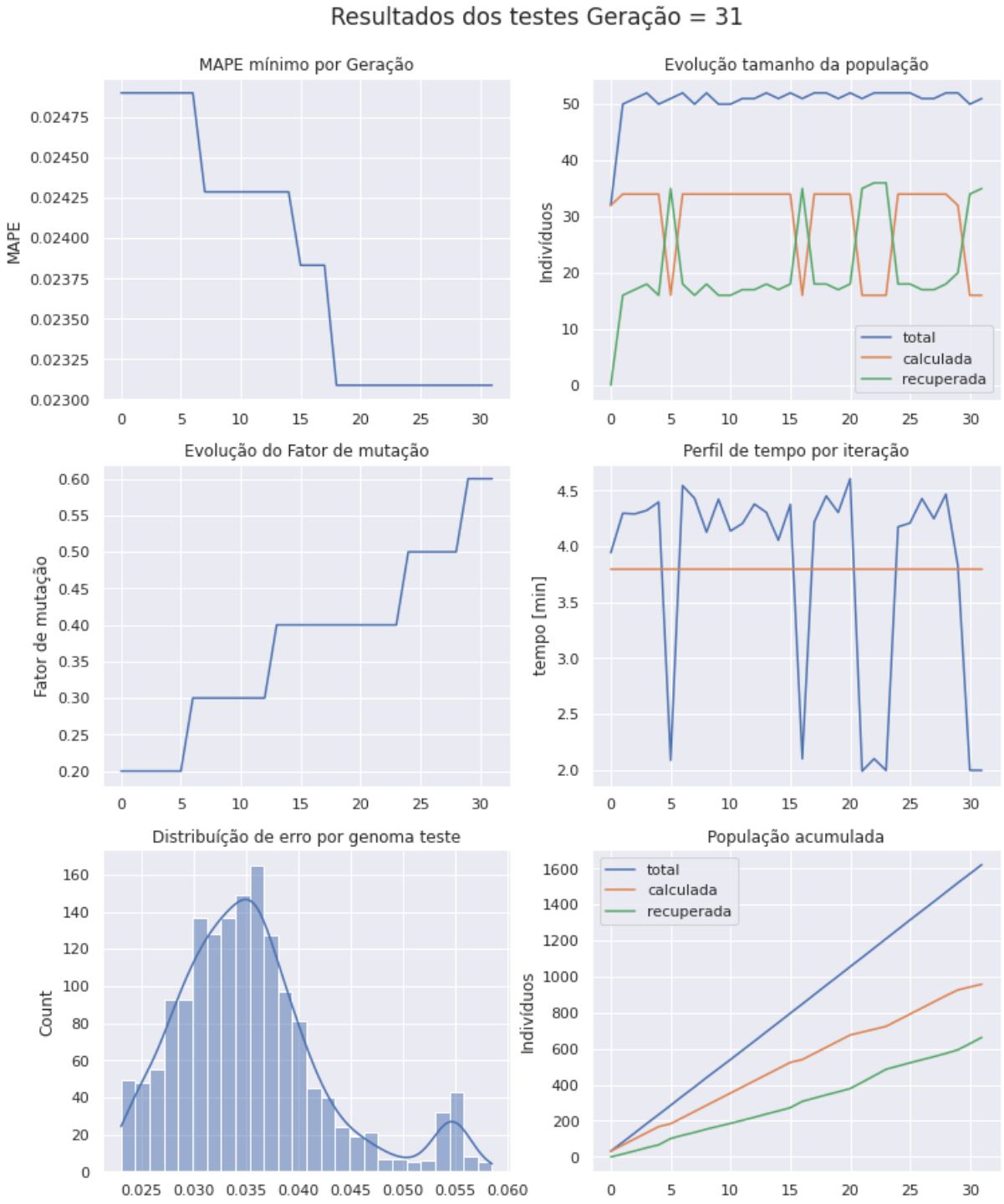
Fonte: Autor

Figura 124 - Resultado da Otimização por Algoritmo Genético (Rural)



Fonte: Autor

Figura 125 - Resultado da Otimização por Algoritmo Genético (Serviço Público)

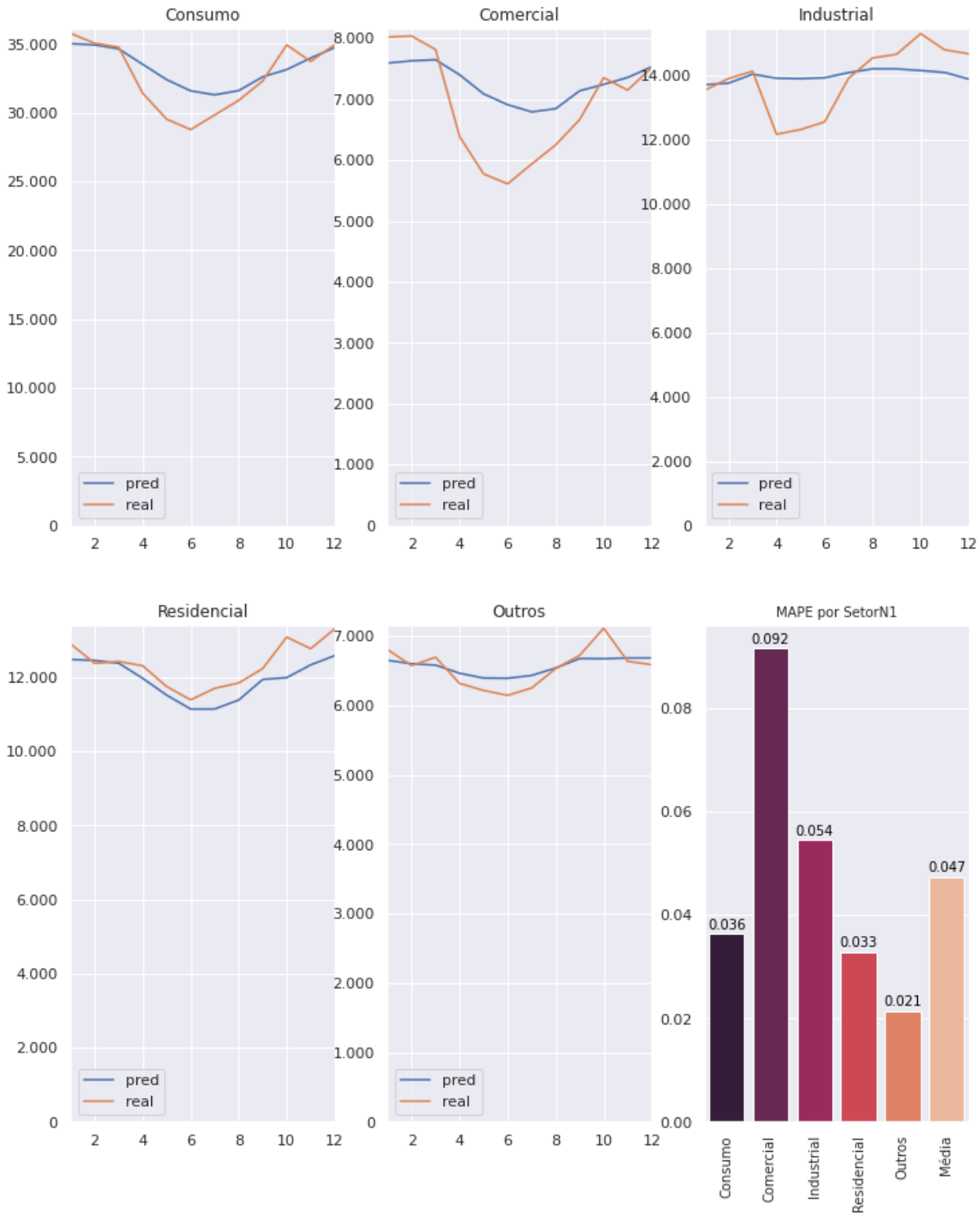


Fonte: Autor

Anexo 7 – Resultados dos 16 cenários de teste com diferentes arranjos de entrada

1. Clima

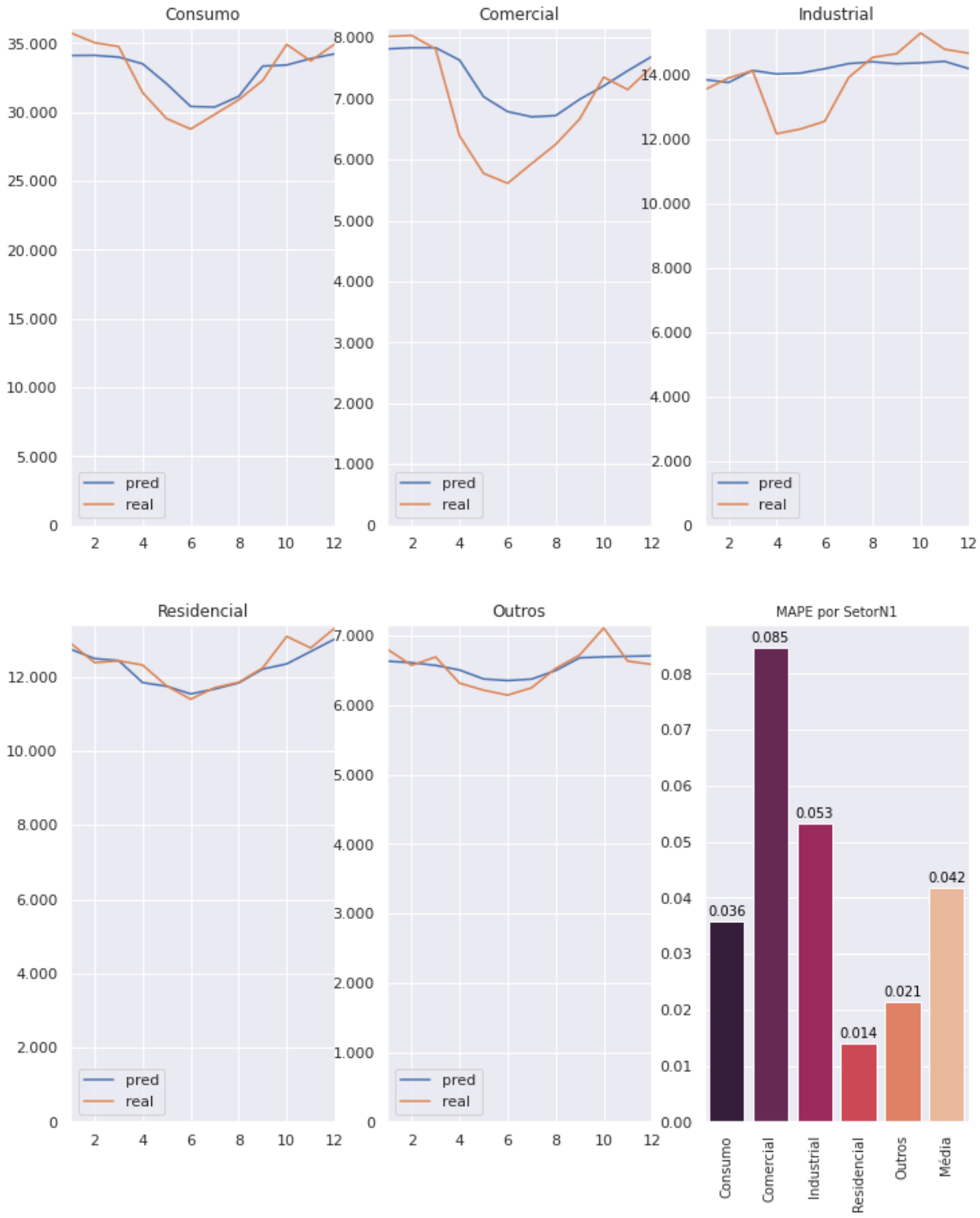
Figura 126 - Teste Valor Real vs Predito e MAPE (BR Resenha)



Fonte: Autor

2. Clima, Óbitos

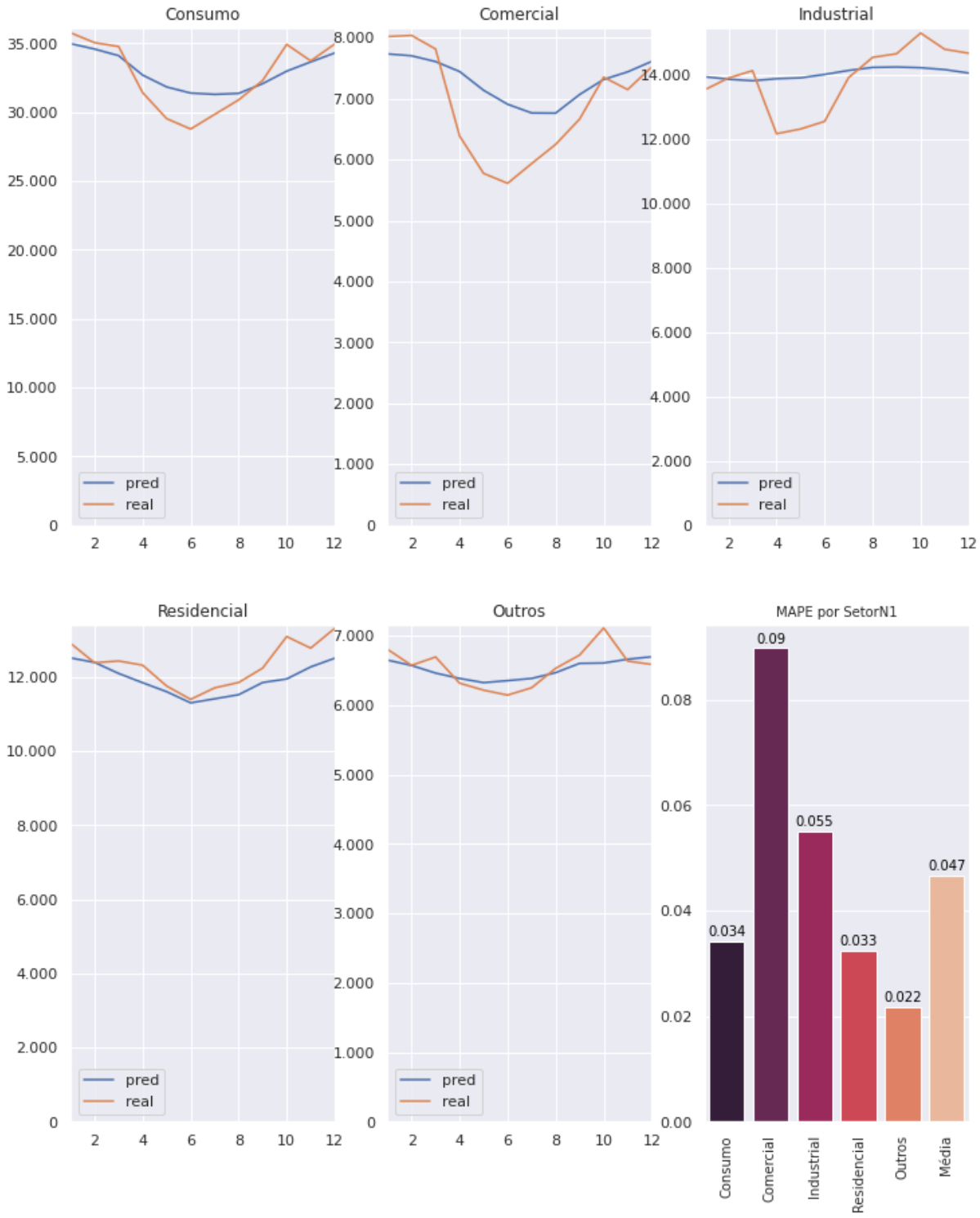
Figura 127 - Teste Valor Real vs Predito e MAPE (BR Resenha, Clima, Óbitos)



Fonte: Autor

3. Clima, IBOV

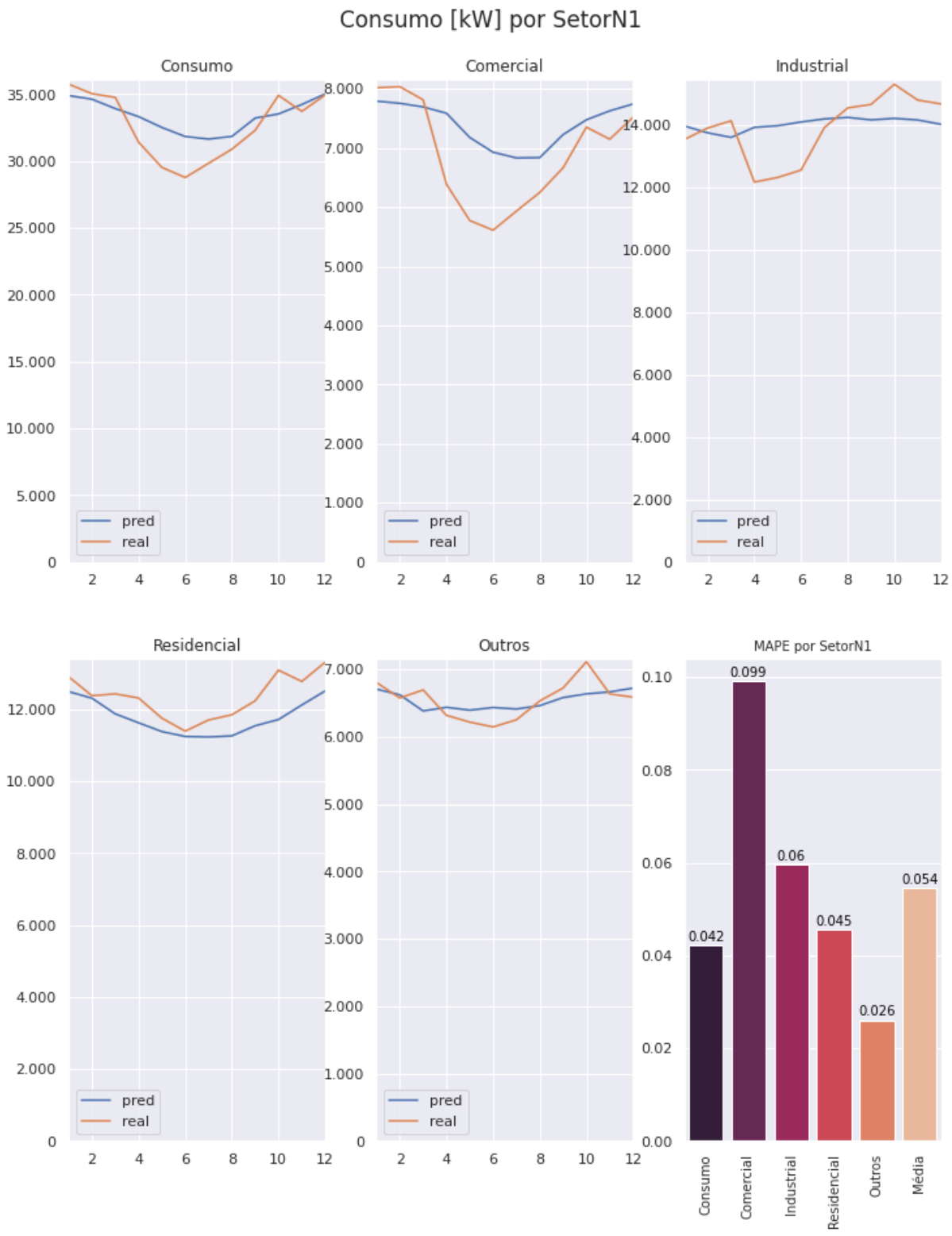
Figura 128 - Teste Valor Real vs Predito e MAPE (BR Resenha, Clima, IBOV)



Fonte: Autor

4. Clima, IBOV, Óbitos

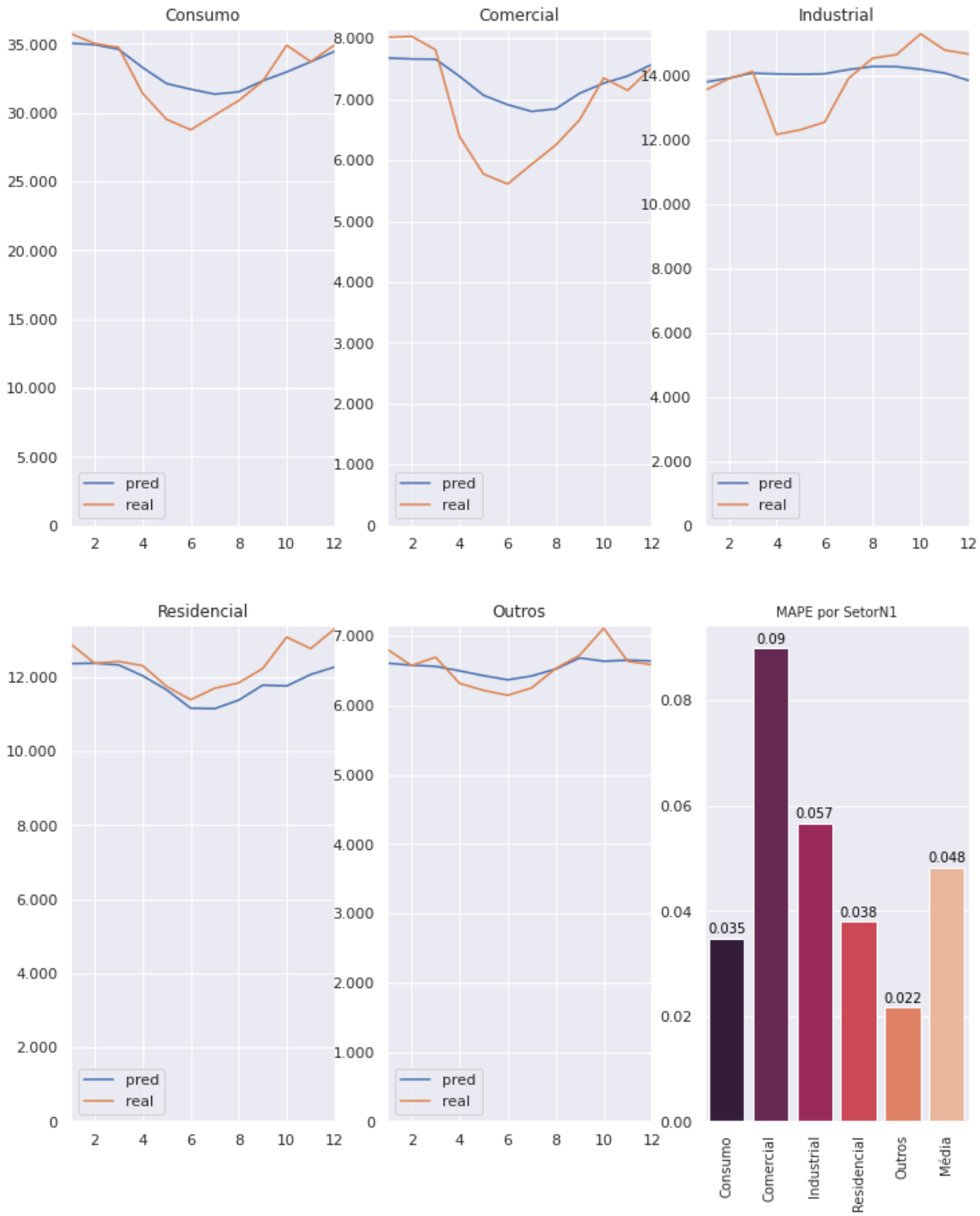
Figura 129 - Teste Valor Real vs Predito e MAPE (BR Resenha, Clima, IBOV, Óbitos)



Fonte: Autor

5. Clima, IPCA

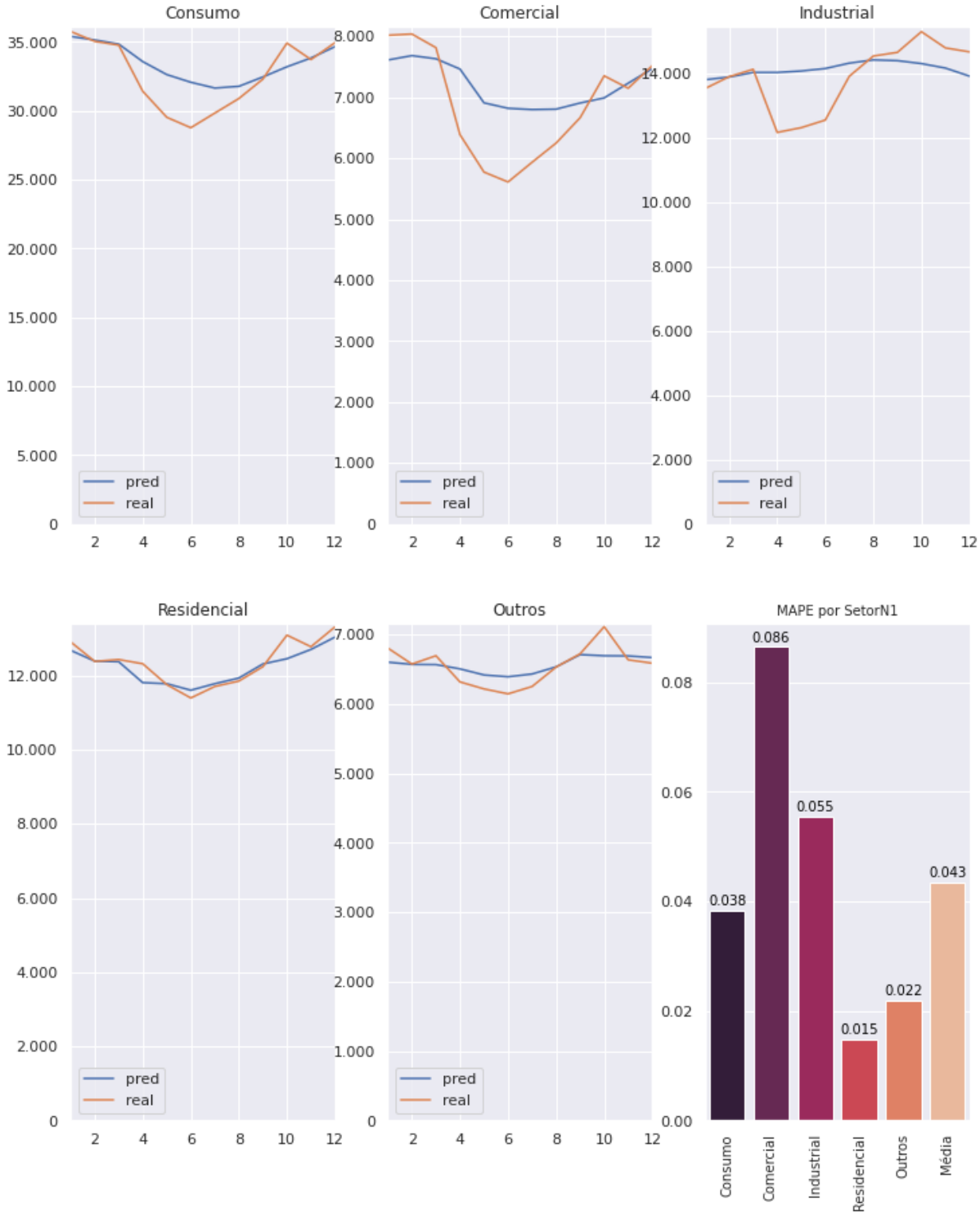
Figura 130 - Teste Valor Real vs Predito e MAPE (BR Resenha, Clima, IPCA)



Fonte: Autor

6. Clima, IPCA, Óbitos

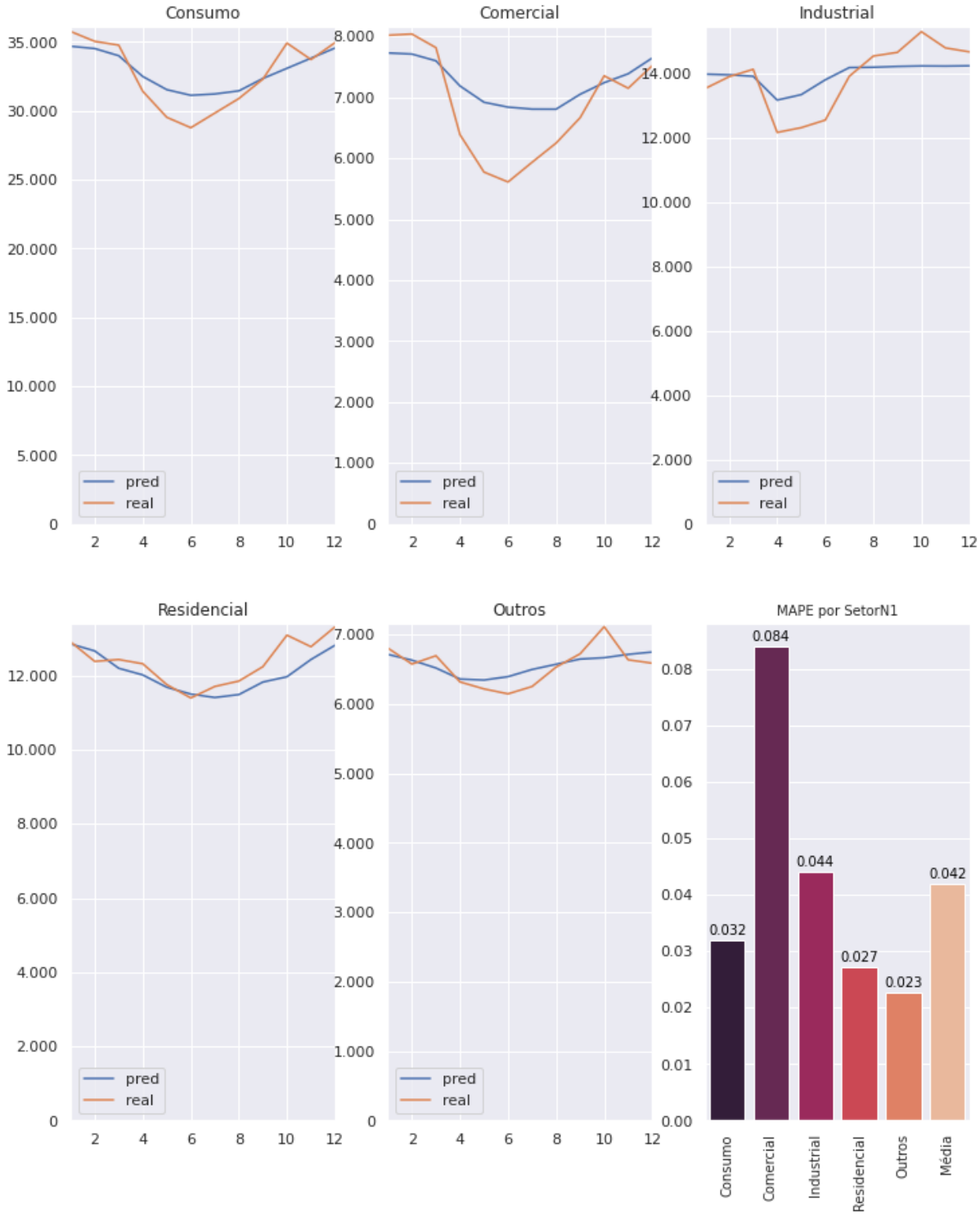
Figura 131 - Teste Valor Real vs Predito e MAPE (BR Resenha, Clima, IPCA, Óbitos)



Fonte: Autor

7. Clima, IBOV, IBCBR

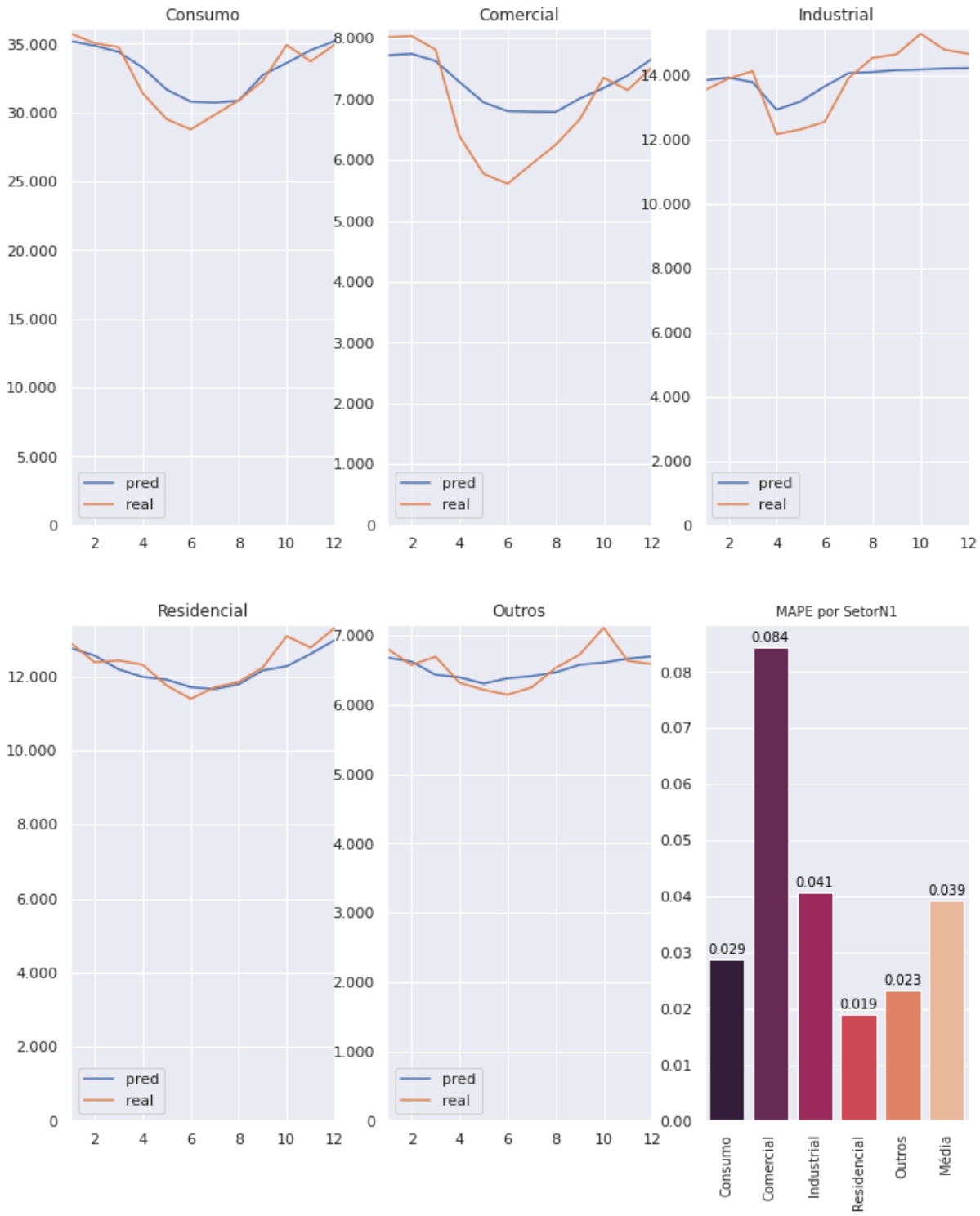
Figura 132 - Teste Valor Real vs Predito e MAPE (BR Resenha, Clima, IBOV, IBCBR)



Fonte: Autor

8. Clima, IBOV, IBCBR, Óbitos

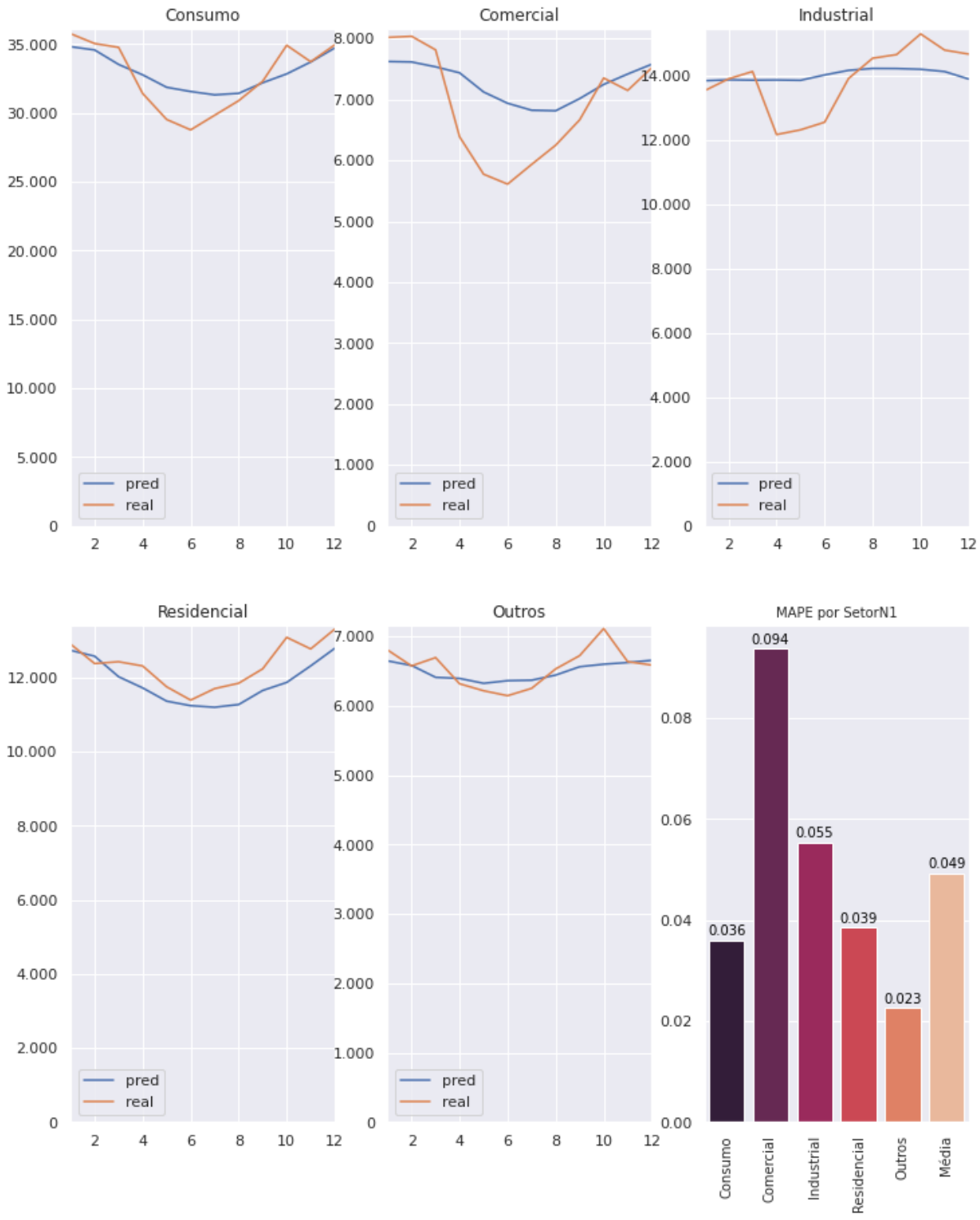
Figura 133 - Teste Valor Real vs Predito e MAPE (BR Resenha, Clima, IBOV, IBCBR, Óbitos)



Fonte: Autor

9. Clima, IBOV, IPCA

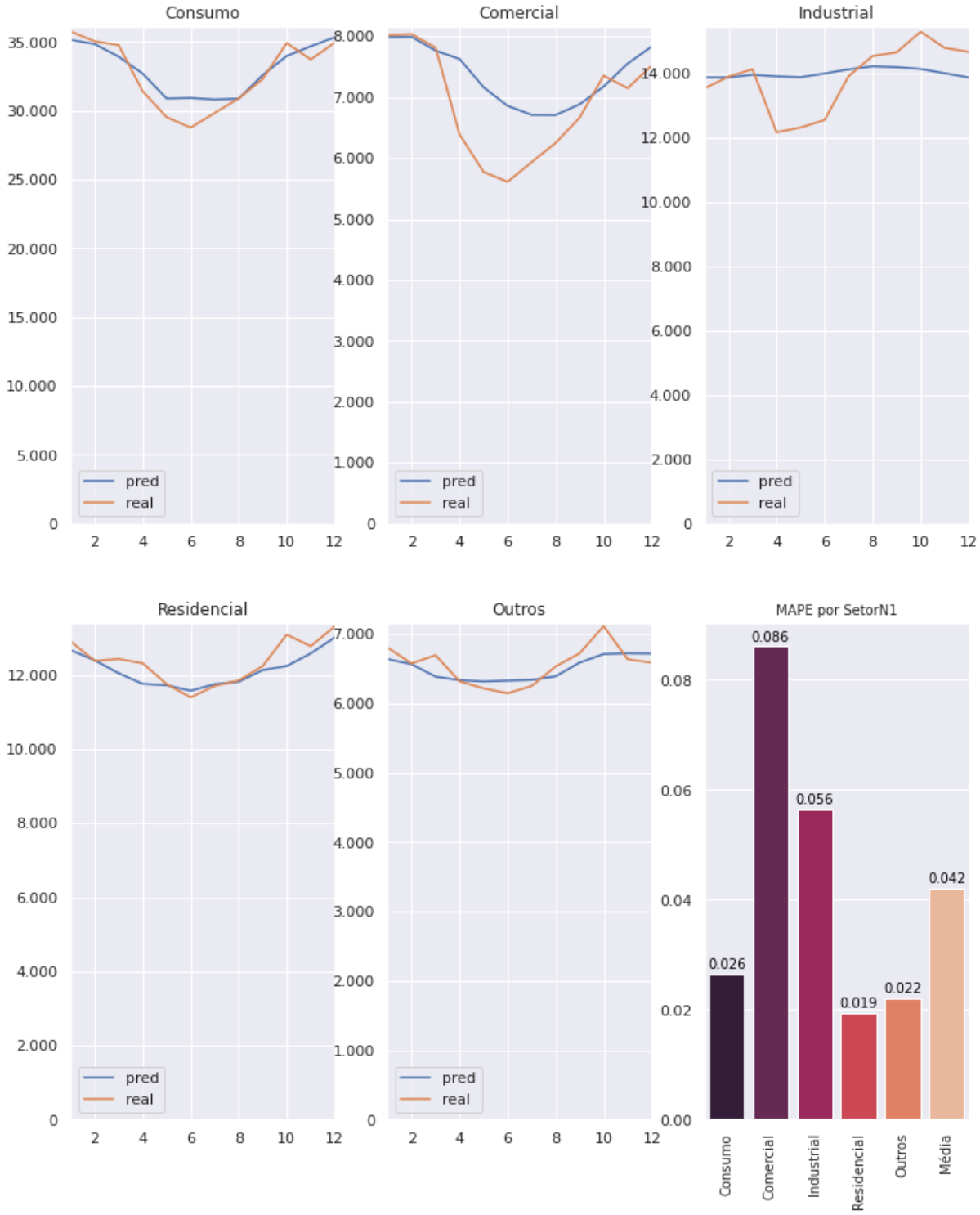
Figura 134 - Teste Valor Real vs Predito e MAPE (BR Resenha, Clima, IBOV, IPCA)



Fonte: Autor

10. Clima, IBOV, IPCA, Óbitos

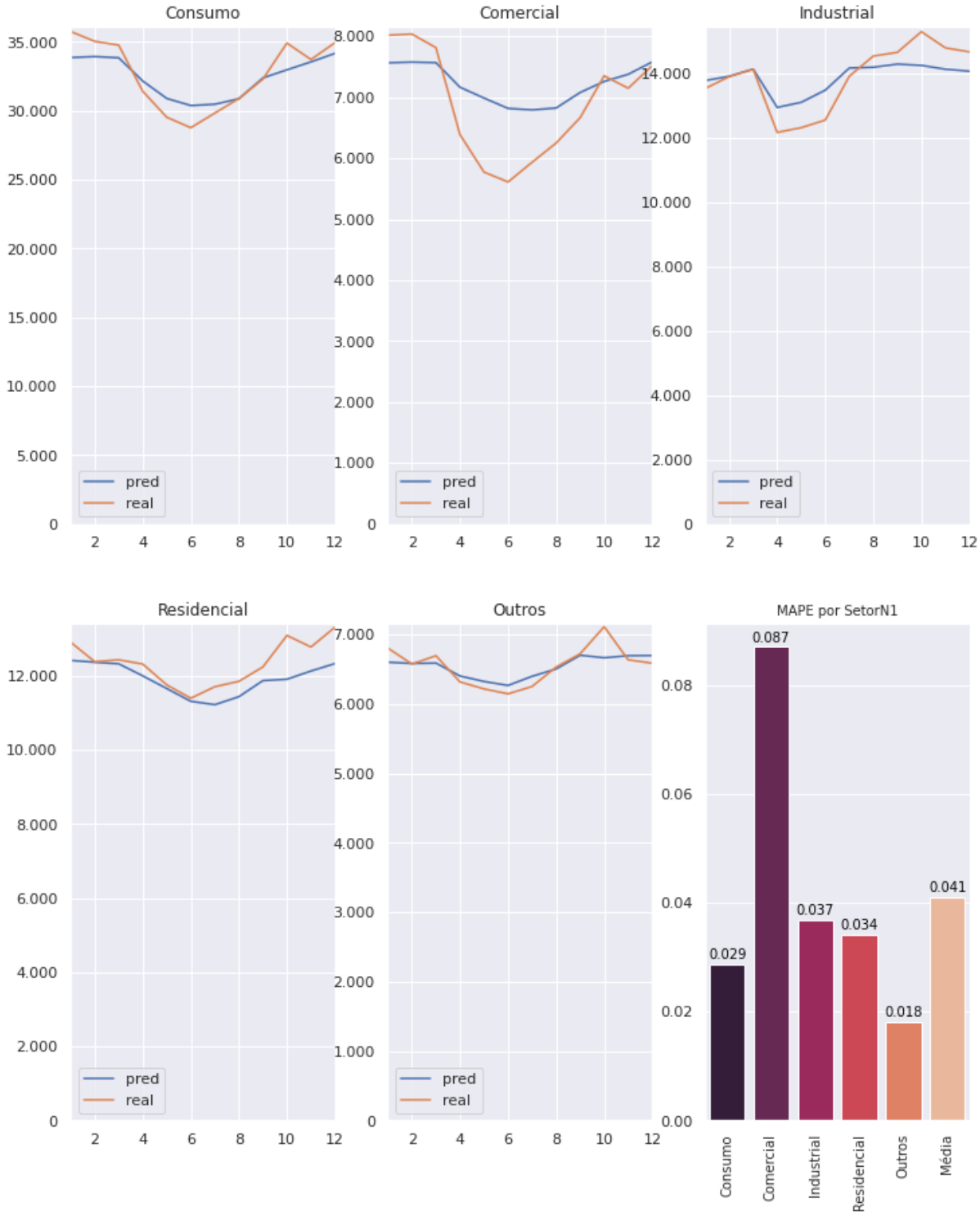
Figura 135 - Teste Valor Real vs Predito e MAPE (BR Resenha, Clima, IBOV, IPCA, Óbitos)



Fonte: Autor

11. Clima, IBCBR, IPCA

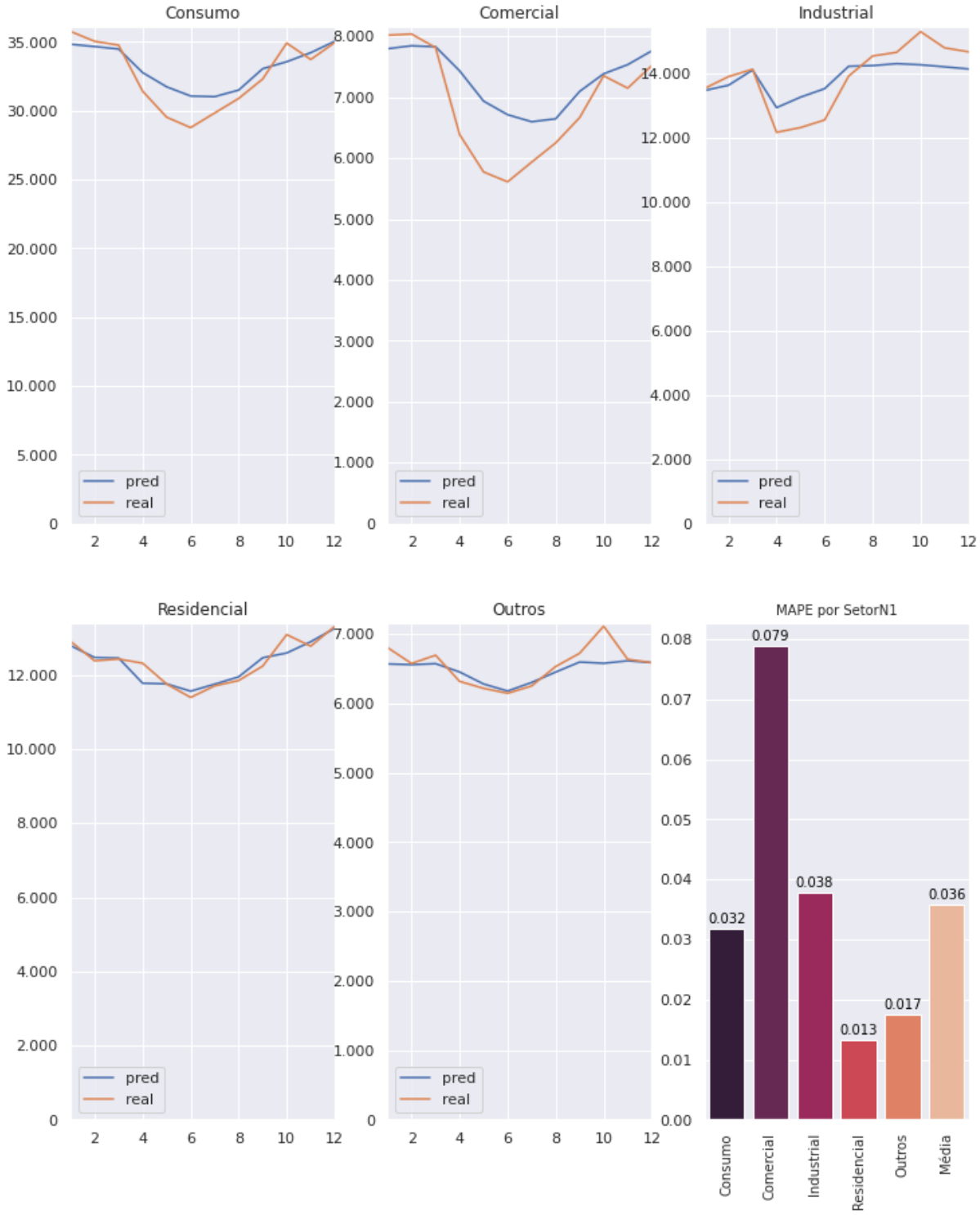
Figura 136 - Teste Valor Real vs Predito e MAPE (BR Resenha, Clima, IBCBR, IPCA)



Fonte: Autor

12. Clima, IBCBR, IPCA, Óbitos

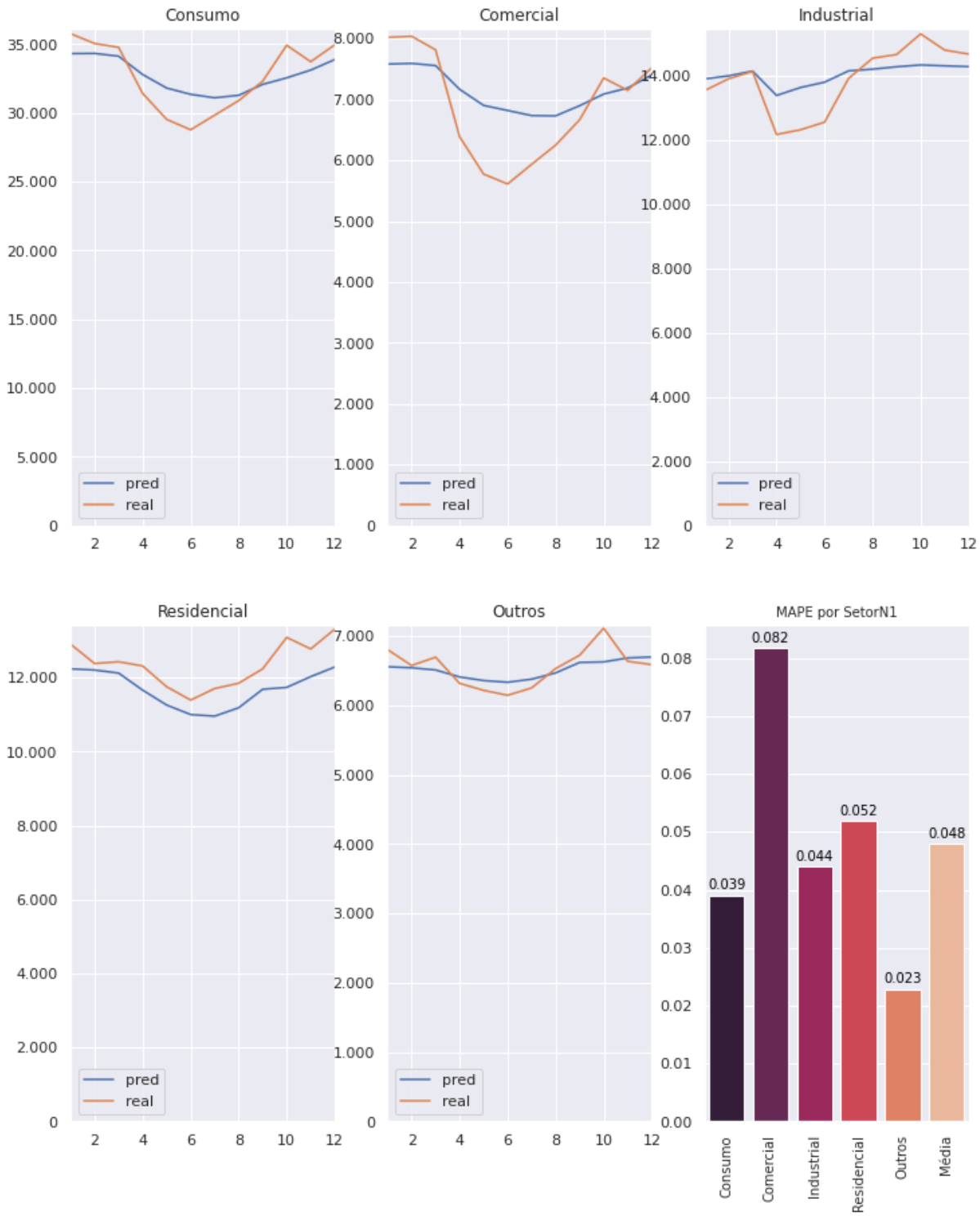
Figura 137 - Teste Valor Real vs Predito e MAPE (BR Resenha, Clima, IBCBR, IPCA, Óbitos)



Fonte: Autor

13. Clima, IBCBR

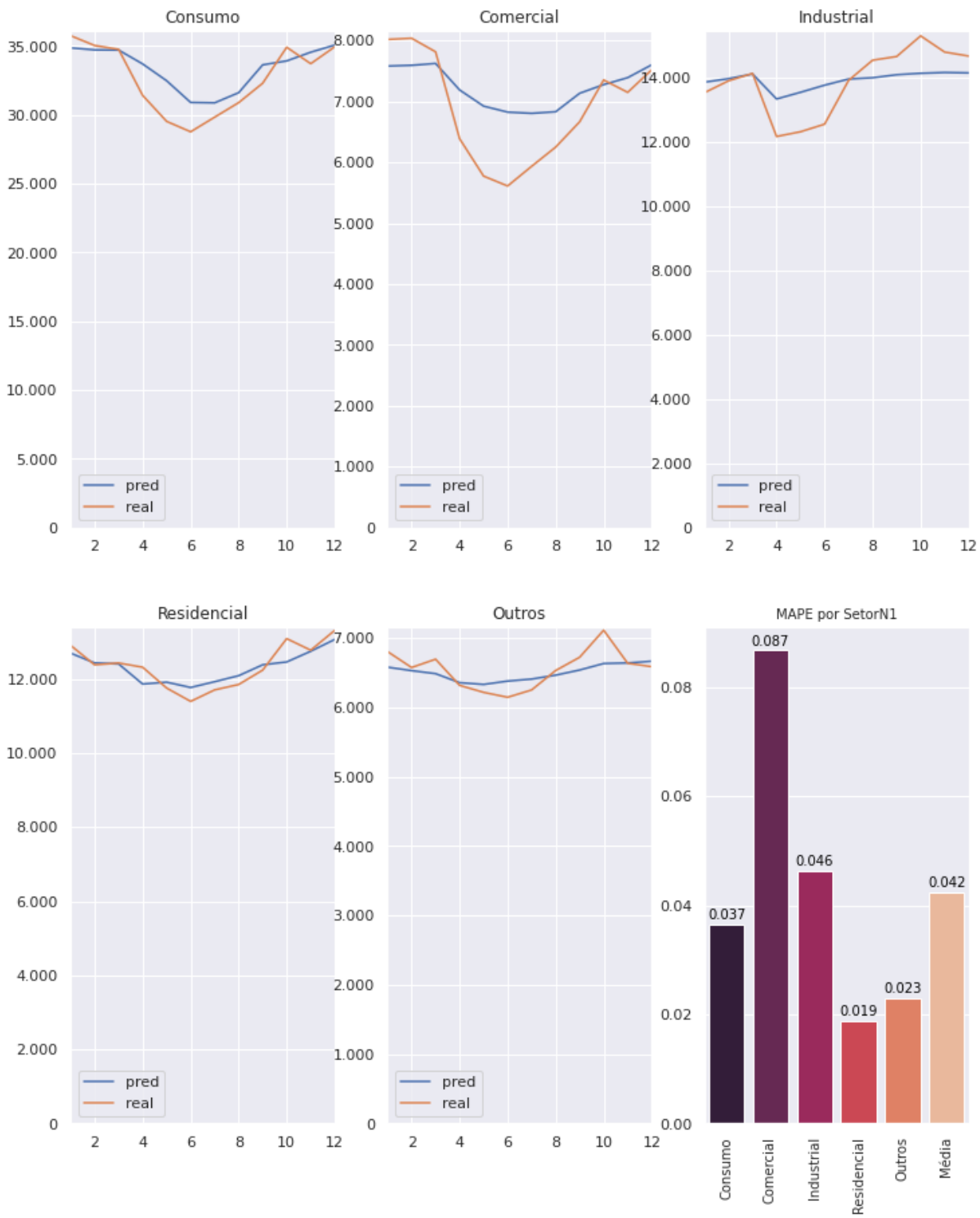
Figura 138 - Teste Valor Real vs Predito e MAPE (BR Resenha, Clima, IBCBR)



Fonte: Autor

14. Clima, IBCBR, Óbitos

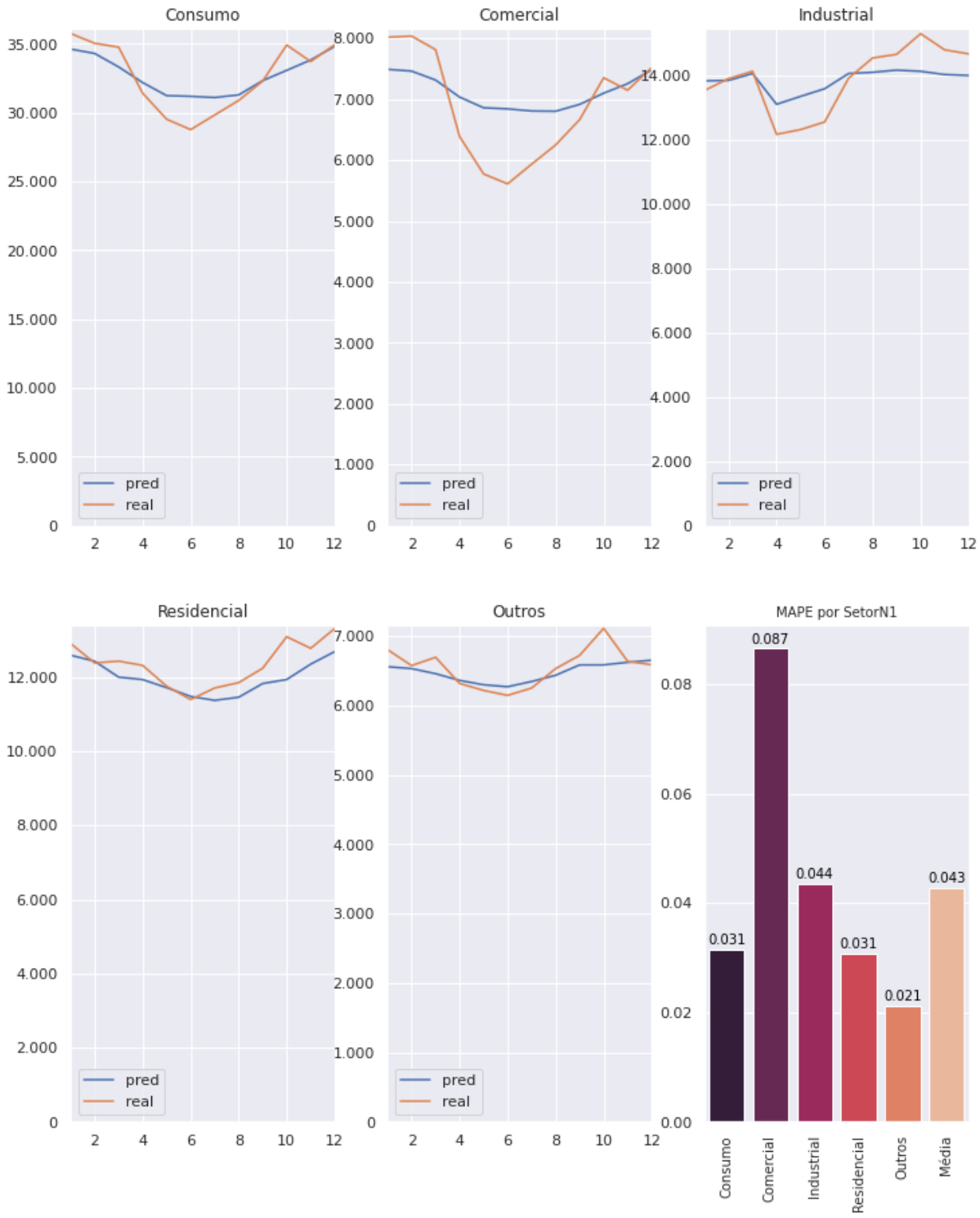
Figura 139 - Teste Valor Real vs Predito e MAPE (BR Resenha, Clima, IBCBR, Óbitos)



Fonte: Autor

15. Clima, IBOV, IBCBR, IPCA

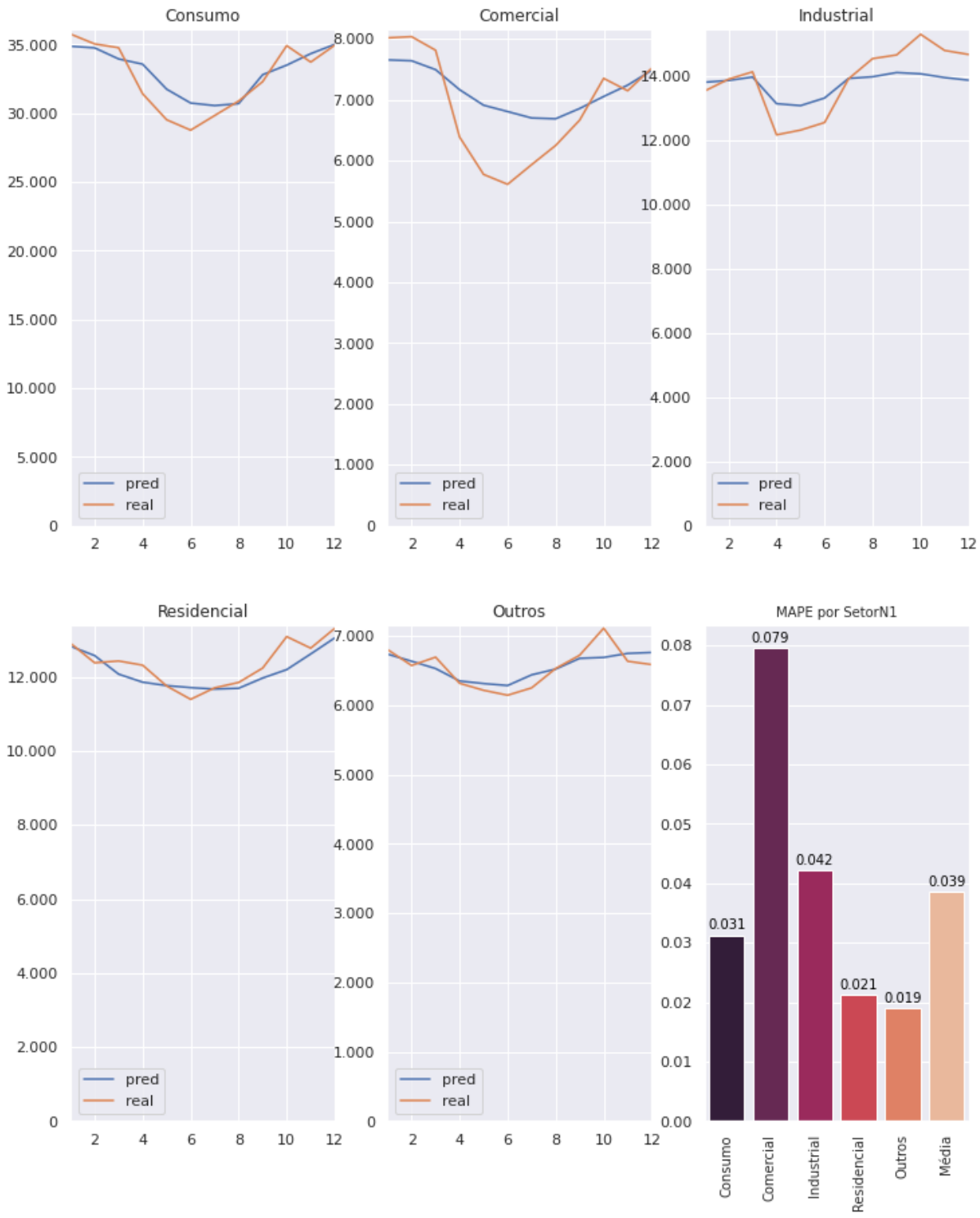
Figura 140 - Teste Valor Real vs Predito e MAPE (BR Resenha, Clima, IBOV, IBCBR, IPCA)



Fonte: Autor

16. Clima, IBOV, IBCBR, IPCA, Óbitos

Figura 141 - Teste Valor Real vs Predito e MAPE (BR Resenha, Clima, IBOV, IBCBR, IPCA, Óbitos)



Fonte: Autor

Anexo 8 – Código Fonte

Todos os códigos desenvolvidos nesse trabalho estão disponíveis em:

https://github.com/lvb86/PD_LSTM_GA/

Uma amostra dos principais estão anexadas abaixo:

Códigos de limpeza e análise de dados:

Elétricos – Resenha EPE

```
"""2004_2021_resenha_EPE.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/github/lvb86/PD_LSTM_GA/blob/main/code/2004_2021_resenha_EPE.ipynb

# Etapa de análise e limpeza de dados

Preparando dados da Resenha Mensal EPE
por: Leandro Barbosa
"""

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

#urlEPE = '/tmp/MERCADO MENSAL PARA DOWNLOAD COLADO.xls'
urlEPE =
'https://docs.google.com/uc?export=download&id=1FtYlt2iaradp5iEebKrWY3mLezRSSqRI'
#https://www.epe.gov.br/pt/publicacoes-dados-abertos/publicacoes/Consumo-mensal-de-energia-eletrica-por-classe-regioes-e-subsistemas

mapmes = {'JAN':'01', 'FEV':'02', 'MAR':'03', 'ABR':'04', 'MAI':'05',
'JUN':'06',
    'JUL':'07', 'AGO':'08', 'SET':'09', 'OUT':'10', 'NOV':'11', 'DEZ':'12'}

lista_setor = ['Residencial', 'Industrial', 'Comercial', 'Outros']

def df_ano_epe( sheet_name, Setor, s):
    df = pd.read_excel(urlEPE,
sheet_name=sheet_name,skiprows=4,usecols=None)
    try:
        df.iloc[0+16*s:1+16*s,:]= int(df.iloc[0+16*s:1+16*s,1].str[0:4])
    except:
        df.iloc[0+16*s:1+16*s,:]= df.iloc[0+16*s:1+16*s,1]
    finally:
        df = df.iloc[0+16*s:9+16*s,:].T.reset_index()
        df.columns =
['old','Ano','Mes','Total','Reg','Norte','Nordeste','Sudeste','Sul','CentoOeste']
        df = df.drop(columns=['old','Reg'])[1:13]
        df.Mes = df.Mes.map(mapmes)
        df['Ano Mês'] = df.Ano.astype(str) +'-'+ df.Mes
        #df.drop(columns = ['Ano', 'Mes'], inplace = True)
```

```
        df['Setor']    = Setor
        df.Ano         = df.Ano.astype(int)
        df.Total       = df.Total.astype(float)
        df.Norte       = df.Norte.astype(float)
        df.Nordeste    = df.Nordeste.astype(float)
        df.Sudeste     = df.Sudeste.astype(float)
        df.Sul         = df.Sul.astype(float)
        df.CentoOeste = df.CentoOeste.astype(float)

        return df[df.Total > 0]

for sheet, setor in enumerate(lista_setor):
    print(sheet, setor)

for sheet, setor in enumerate(lista_setor):
    for s in range(18):
        if sheet == 0 and s ==0:
            dfEPE = df_ano_epe( sheet+1,setor,s)
        else:
            dfEPE = dfEPE.append(df_ano_epe( sheet+1,setor,s))
dfEPE

import seaborn as sns
import matplotlib.pyplot as plt

for e, i in enumerate(lista_setor):
    plt.figure(figsize=(10,8))
    sns.lineplot(data = dfEPE.query(f'Setor == "{i}"'),x = 'Mes', y =
'Total', hue = 'Ano' )
    plt.title(i, fontsize = 17)

dfEPE.to_csv('/tmp/ResenhaEPE.csv', index=False)

dfEPE[(dfEPE.Ano == 2020) & (dfEPE.Mes == '01')]
```

Climáticos

```
"""BR_Clima_2013_2020.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/github/lvb86/PD_LSTM_GA/blob/main/code/BR_Clima_2013_2020.ipynb

#Etapa de análise e limpeza de dados

Extração de DF Clima para scenário BR
por Leandro Barbosa

##Declarações
"""

pip install -U scikit-learn

# Commented out IPython magic to ensure Python compatibility.
import math
import matplotlib.pyplot    as plt
import matplotlib.ticker    as ticker
import matplotlib.style     as style
import numpy                as np
import seaborn              as sns
import pandas               as pd

from urllib.request                 import urlopen
from zipfile                        import ZipFile
from statsmodels.graphics.tsaplots  import plot_acf, plot_pacf
from statsmodels.tsa.seasonal       import seasonal_decompose
from sklearn.metrics                import mean_squared_error
from sklearn.metrics                import mean_absolute_percentage_error
from sklearn.linear_model           import LinearRegression

stl = style.available
# %matplotlib inline

swap     = '/tmp/'
prefGo   = 'https://drive.google.com/u/0/uc?export=download&id='
urlINMET = prefGo + '1FNrp311pEegFczI7NWCCXqG5BVPSHd6e'

mes =
['Jan','Fev','Mar','Abr','Mai','Jun','Jul','Ago','Set','Out','Nov','Dez']
mesN = [1,2,3,4,5,6,7,8,9,10,11,12]

palettes = ['Accent', 'Accent_r', 'Blues', 'Blues_r', 'BrBG', 'BrBG_r',
            'BuGn', 'BuGn_r', 'BuPu', 'BuPu_r', 'CMRmap', 'CMRmap_r',
            'Dark2', 'Dark2_r', 'GnBu', 'GnBu_r', 'Greens', 'Greens_r',
            'Greys', 'Greys_r', 'OrRd', 'OrRd_r', 'Oranges', 'Oranges_r',
            'PRGn', 'PRGn_r', 'Paired', 'Paired_r', 'Pastel1', 'Pastel1_r',
            'Pastel2', 'Pastel2_r', 'PiYG', 'PiYG_r', 'PuBu', 'PuBuGn',
            'PuBuGn_r', 'PuBu_r', 'PuOr', 'PuOr_r', 'PuRd', 'PuRd_r',
            'Purples', 'Purples_r', 'RdBu', 'RdBu_r', 'RdGy', 'RdGy_r',
            'RdPu', 'RdPu_r', 'RdYlBu', 'RdYlBu_r', 'RdYlGn', 'RdYlGn_r',
            'Reds', 'Reds_r', 'Set1', 'Set1_r', 'Set2', 'Set2_r', 'Set3',
            'Set3_r', 'Spectral', 'Spectral_r',
```

```
            #'Vega10', 'Vega10_r', 'Vega20',
            'Vega20_r', 'Vega20b', 'Vega20b_r', 'Vega20c', 'Vega20c_r',
            'Wistia', 'Wistia_r', 'YlGn', 'YlGnBu', 'YlGnBu_r', 'YlGn_r',
            'YlOrBr', 'YlOrBr_r', 'YlOrRd', 'YlOrRd_r', 'afmhot',
'afmhot_r',
            'autumn', 'autumn_r', 'binary', 'binary_r', 'bone', 'bone_r',
            'brg', 'brg_r', 'bwr', 'bwr_r', 'cool', 'cool_r', 'coolwarm',
            'coolwarm_r', 'copper', 'copper_r', 'cubehelix', 'cubehelix_r',
            'flag', 'flag_r', 'gist_earth', 'gist_earth_r', 'gist_gray',
            'gist_gray_r', 'gist_heat', 'gist_heat_r', 'gist_ncar',
            'gist_ncar_r', 'gist_rainbow', 'gist_rainbow_r', 'gist_stern',
            'gist_stern_r', 'gist_yarg', 'gist_yarg_r', 'gnuplot',
'gnuplot2',
            'gnuplot2_r', 'gnuplot_r', 'gray', 'gray_r', 'hot', 'hot_r',
'hsv',
            'hsv_r', 'icefire', 'icefire_r', 'inferno', 'inferno_r',
            #'jet', ##'jet_r',
            'magma', 'magma_r', 'mako', 'mako_r', 'nipy_spectral',
            'nipy_spectral_r', 'ocean', 'ocean_r', 'pink', 'pink_r',
'plasma',
            'plasma_r', 'prism', 'prism_r', 'rainbow', 'rainbow_r',
'rocket',
            'rocket_r', 'seismic', 'seismic_r', ##'spectral', 'spectral_r',
            'spring', 'spring_r', 'summer', 'summer_r', 'tab10', 'tab10_r',
            'tab20', 'tab20_r', 'tab20b', 'tab20b_r', 'tab20c', 'tab20c_r',
            'terrain', 'terrain_r', 'viridis', 'viridis_r', 'vlag',
'vlag_r',
            'winter', 'winter_r']

# Definições de estilo
if 1==1: #tema Claro para artigo retrato
    DefPalette = palettes[13] #13 Dark2
    DefPaletteHist = 'rocket'
    DefStyle = stl[15] #15 Seaborn-Darkgrid
    DefSize = (10,6) #(17,6)
    DefGrid = ('-')

else: # tema Escuro para apresentação paisagem
    DefPalette = palettes[123] #123 - hsv
    DefPaletteHist = 'rocket'
    DefStyle = stl[4] #15 dark_background
    DefSize = (10,6) #(17,6)
    DefGrid = (':')
    plt.rc_context({'axes.edgecolor':'gray', 'xtick.color':'white',
                    'ytick.color':'white', 'figure.facecolor':'black'})

# Para traduzir ticker para PT-BR
def formatador_de_milhares(valor, p):
    valor = f"{valor:,.0f}"
    mapa_de_traducao = str.maketrans(',.', '.,')
    return valor.translate(mapa_de_traducao)
#exemplo de uso
#ax.yaxis.set_major_formatter(ticker.FuncFormatter(formatador_de_milhares))
#Powered by @lpeixoto2000

"""## Análise de Consumo de Energia

### Importação dos dados

## Dados INMET
"""

zipresp = urlopen(urlINMET)                 # Download from URL
```

```
tempzip = open("/tmp/tempfile.zip", "wb")   # Create a new file
tempzip.write(zipresp.read())               # Write the contents of the
downloaded file into the new file
tempzip.close()                             # Close the newly-created file
zip = ZipFile("/tmp/tempfile.zip")          # Re-open the newly-created file
with ZipFile()

#Adaptado de
#https://svaderia.github.io/articles/downloading-and-unzipping-a-zipfile/

#zip.infolist()
listINMET = zip.namelist()

zip.namelist()[0]

zip.namelist()[1]

pd.read_csv(zip.open(zip.namelist()[1]),encoding='UTF-8', skiprows=9,
sep=';',parse_dates=True, decimal=',').head(3)

pd.read_csv(zip.open(zip.namelist()[1]),encoding='UTF-8', nrows=8,
sep=';',parse_dates=True, decimal=',').head(3)

zip.namelist()[1][-32:-28]

for e,i in enumerate(listINMET):
    #print(e,i)
    if e ==0:
        pass
    elif e == 1:
        climaD = pd.read_csv(zip.open(i),encoding='UTF-8', skiprows=9,
sep=';',parse_dates=True, decimal=',')
        climaD['origem'] = i[-32:-28]
    else:
        climaD2 = pd.read_csv(zip.open(i),encoding='UTF-8', skiprows=9,
sep=';',parse_dates=True, decimal=',')
        climaD2['origem'] = i[-32:-28]
        climaD = climaD.append(climaD2)
#clima.sample(5)
climaD.shape

climaD.head(3)

#Carregando todos cabeçalhos dos arquivos INMET
for e, i in enumerate(listINMET):
    if e ==0:
        pass

    elif e ==1:
        climaH = pd.read_csv(zip.open(i), encoding='UTF-8', nrows=8, sep=':
',
                             parse_dates=True, decimal=',').T
        header = climaH.iloc[0]
        climaH.columns = header
        climaH = climaH[1:]
    else:
        climaH2 = pd.read_csv(zip.open(i), encoding='UTF-8', nrows=8,
sep=': ',
                              parse_dates=True, decimal=',').T
        climaH2.columns = header
        climaH2 = climaH2[1:]
        climaH = climaH.append(climaH2)
climaH.shape
```

```
climaH.head(3)

climaH.Situacao.value_counts()

climaH = climaH.reset_index().set_index('Codigo
Estacao').rename(columns={'index':'Cidade','Codigo Estacao':'origem'})

climaH.head(3)

climaD = climaD.fillna(method='ffill')

climaD.head(3)

colunasNomes = {
    'Data Medicao':'data',
    'NUMERO DE DIAS COM PRECIP. PLUV, MENSAL
(AUT)(número)':'Dias_com_precip',
    'PRECIPITACAO TOTAL, MENSAL (AUT)(mm)':'precipitacao',
    'PRESSAO ATMOSFERICA, MEDIA MENSAL (AUT)(mB)':'pressao',
    'TEMPERATURA MEDIA, MENSAL (AUT)(°C)':'temperatura',
    'VENTO, VELOCIDADE MAXIMA MENSAL (AUT)(m/s)':'vento_max',
    'VENTO, VELOCIDADE MEDIA MENSAL (AUT)(m/s)':'vento_med',
    'Unnamed: 7':'un',
    'origem':'origem'
}
climaD.rename(columns=colunasNomes, inplace = True)
climaD.drop(columns=['un'], inplace = True)
climaD.columns

#Cabeçalhos
climaH.head(3)

clima = pd.merge(climaD,climaH, left_on='origem', right_on='Codigo
Estacao', how='right')

clima.head(3)

clima.Situacao.value_counts()

# Excluindo Estações em Pane
clima = clima.query("Situacao != 'Pane' ")

parametros = climaD.columns[1:-1]
parametros

clima[['data','temperatura']].groupby('data').mean()

Mediatemperatura = clima[['data','temperatura']].groupby('data').mean()
Mediatemperatura.rename(columns={'temperatura':('temperatura','media')},
inplace = True)

MediaDias_com_precip =
clima[['data','Dias_com_precip']].groupby('data').mean()
MediaDias_com_precip.rename(columns={'Dias_com_precip':('Dias_com_precip','
media')}, inplace = True)

Mediaprecipitacao = clima[['data','precipitacao']].groupby('data').mean()
Mediaprecipitacao.rename(columns={'precipitacao':('precipitacao','media')},
inplace = True)

Mediapressao = clima[['data','pressao']].groupby('data').mean()
```

```
Mediapressao.rename(columns={'pressao':('pressao','media')}, inplace =
True)

Mediavento_max = clima[['data','vento_max']].groupby('data').mean()
Mediavento_max.rename(columns={'vento_max':('vento_max','media')}, inplace
= True)

Mediavento_med = clima[['data','vento_med']].groupby('data').mean()
Mediavento_med.rename(columns={'vento_med':('vento_med','media')}, inplace
= True)

clima_med=clima[['data','Cidade',*parametros]].groupby(['data','Cidade']).m
ean()

#clima[['data','temperatura','Cidade']].pivot(['Cidade'])
#.groupby(['data']).mean()#.pivot('data','Cidade').head(3)

climaPivot =
clima_med.reset_index()[['data','temperatura','Cidade']].pivot('data','Cida
de')
climaPivot = pd.merge(climaPivot, Mediatemperatura, right_on='data',
left_on='data')

climaPivot2 =
clima_med.reset_index()[['data','Dias_com_precip','Cidade']].pivot('data','
Cidade')
climaPivot2 = pd.merge(climaPivot2, MediaDias_com_precip, right_on='data',
left_on='data')
climaPivot = pd.merge(climaPivot, climaPivot2, right_on='data',
left_on='data')

climaPivot2 =
clima_med.reset_index()[['data','precipitacao','Cidade']].pivot('data','Cid
ade')
climaPivot2 = pd.merge(climaPivot2, Mediaprecipitacao, right_on='data',
left_on='data')
climaPivot = pd.merge(climaPivot, climaPivot2, right_on='data',
left_on='data')

climaPivot2 =
clima_med.reset_index()[['data','pressao','Cidade']].pivot('data','Cidade')
climaPivot2 = pd.merge(climaPivot2, Mediapressao, right_on='data',
left_on='data')
climaPivot = pd.merge(climaPivot, climaPivot2, right_on='data',
left_on='data')

climaPivot2 =
clima_med.reset_index()[['data','vento_max','Cidade']].pivot('data','Cidade
')
climaPivot2 = pd.merge(climaPivot2, Mediavento_max, right_on='data',
left_on='data')
climaPivot = pd.merge(climaPivot, climaPivot2, right_on='data',
left_on='data')

climaPivot2 =
clima_med.reset_index()[['data','vento_med','Cidade']].pivot('data','Cidade
')
climaPivot2 = pd.merge(climaPivot2, Mediavento_med, right_on='data',
left_on='data')
climaPivot = pd.merge(climaPivot, climaPivot2, right_on='data',
left_on='data')

climaPivot.head()
```

```
def analise_correlacao(par):
    '''
    Fitra prarâmetro no data frame para gerar mapa de correlação e ranking
    par: entrada de parâmento
    return mapa de correlação e ranking
    '''

    display(par)
    df = climaPivot[par]
    #display(df.head())
    #display(df.corr())

    plt.figure(figsize=(30,30))
    ax = sns.heatmap(df.corr(),cmap="YlGnBu", annot=True)
    plt.title('Correlação Cruzada de '+ par,fontsize=20,y=1.10)
    plt.show();

    ranking = climaPivot[par].corr().sum()
    ranking.sort_values(ascending=False, inplace=True)
    rangingCorrMed = ranking/len(ranking)
    #sns.barplot(rangingCorrMed.reset_index())
    plt.figure(figsize=DefSize)
    ax = rangingCorrMed[:10].plot(kind='bar')
    plt.title('Ranking top 10 maiores correlações médias de '+
par,fontsize=17,y=1.05)
    plt.ylim(rangingCorrMed[10],rangingCorrMed.max())
    plt.show();

if 0:
    for i in parametros:
        analise_correlacao(i)

d = {('dt','data'): climaPivot.reset_index()['data'],
     ('dt','Ano'): climaPivot.reset_index()['data'].str[0:4],
     ('dt','Mês'): climaPivot.reset_index()['data'].str[5:7],
     ('dt','Ano Mês'): climaPivot.reset_index()['data'].str[0:7]}
datas = pd.DataFrame(d)

climaPivot =
pd.merge(climaPivot,datas['dt'],right_on='data',left_on=('data'))
climaPivot.rename(columns={'Ano':('dt','Ano'),'Mês':('dt','Mês'),'Ano
Mês':('dt','Ano Mês')}, inplace = True)

climaPivot.head()

parametrosT = ['Dias com Precipitação', 'Precipitacão', 'Pressão',
'Temperatura',
       'Vento Maxímo', 'Vento Médio']

parametrosU = ['', '[mm]', '[mB]', '[°C]',
       '[m/s]', '[m/s]']

len(parametros)

climaPivot.sample(3)

def perfil_medio_clima(_i, _Prefixo = 'Perfil Médio de ', _Escopo = ' no
BR'):
    '''
    Gera gráficos de perfis de históricos climáticos apartir de Dataframe e
     Parâmetros de entrada
    '''
```

```
    dfTemp = climaPivot[[('dt', 'Ano'),('dt', 'Mês'),('dt', 'Ano
Mês'),(parametros[_i],'media')]].reset_index()
    dfTemp.columns = ['col1','Ano','Mês','Ano Mês',parametros[_i]]
    dfTemp.drop('col1', axis =1, inplace = True)
    sns.set_theme()
    style.use(DefStyle)
    plt.figure(figsize=DefSize)
    plt.title(_Prefixo + parametrosT[_i] + _Escopo,y=1.05,fontsize = 17)
    ax = sns.lineplot(data=dfTemp,palette=DefPalette, x = 'Mês', y =
parametros[_i], hue = 'Ano') #'hsv'
    plt.grid(True,linestyle = ':')
    #plt.xticks(rotation = 30)

    ax.legend(loc='center right', bbox_to_anchor=(1.15, 0.5), ncol=1,
fancybox=False, shadow=True, title = 'Ano')
    plt.ylabel(parametrosT[_i] + ' ' + parametrosU[_i])
    plt.xlabel('Mês')
    plt.show()

for i in range(0,len(parametros)):
    perfil_medio_clima(i, 'Perfil Médio de ', ' no BR')

"""##Juntando DF"""

climaPivot.sample(5)

dfClimaMed = climaPivot[[('data'),('dt', 'Ano'),('dt', 'Mês'),('dt', 'Ano
Mês'),('Dias_com_precip','media'),

('precipitacao','media'),('pressao','media'),('temperatura','media'),
            ('vento_max','media'),('vento_med','media')]].reset_index()

dfClimaMed.columns = ['col1','Data','Ano','Mês','Ano Mês','Dias com
Precipitação',
                  'Precipitação','Pressão','Temperatura','Vento Máx','Vento
Méd']
dfClimaMed.drop('col1', axis =1, inplace = True)
dfClimaMed.head()

VarClima = dfClimaMed.corr().columns

dfClimaMed.corr()

sns.set_theme()
style.use(DefStyle)  #4
#plt.rc_context({'axes.edgecolor':'gray', 'xtick.color':'white',
'axes.titlecolor':'white','ytick.color':'white',
'figure.facecolor':'black'})

plt.figure(figsize=DefSize)
plt.title('Correlação ente as variáveis',y=1.05,fontsize = 17)
ax = sns.heatmap(dfClimaMed.corr(),cmap="YlGnBu", annot=True)

dfClimaMed['Data'] = pd.to_datetime(dfClimaMed['Data'])

dfClimaMed.head()

dfClimaMed.columns

VarClima

dfClimaMed.to_csv(swap+'BR_2013_2020_clima.csv', index=False)
```

```
dfClimaMed
```

IBOV

```
"""IBOV.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/github/lvb86/PD_LSTM_GA/blob/main/code/IBOV.ipynb

# Etapa de análise e limpeza de dados

Preparando dados IBC-BR
por: Leandro Barbosa
"""

# Commented out IPython magic to ensure Python compatibility.
import math
import matplotlib.pyplot    as plt
import matplotlib.ticker    as ticker
import matplotlib.style     as style
import numpy                as np
import seaborn              as sns
import pandas               as pd

from urllib.request                 import urlopen
from zipfile                        import ZipFile
from statsmodels.graphics.tsaplots  import plot_acf, plot_pacf
from statsmodels.tsa.seasonal       import seasonal_decompose
from sklearn.metrics                import mean_squared_error
#from sklearn.metrics                 import mean_absolute_percentage_error
from sklearn.linear_model           import LinearRegression

stl = style.available
# %matplotlib inline

prefGo  = 'https://docs.google.com/uc?export=download&id='
urlibov = prefGo + '1S2SQ98qk3V52LnYde04QR2k-X8VhZ_n6'

mes =
['Jan','Fev','Mar','Abr','Mai','Jun','Jul','Ago','Set','Out','Nov','Dez']
mesN = [1,2,3,4,5,6,7,8,9,10,11,12]

palettes = ['Accent', 'Accent_r', 'Blues', 'Blues_r', 'BrBG', 'BrBG_r',
            'BuGn', 'BuGn_r', 'BuPu', 'BuPu_r', 'CMRmap', 'CMRmap_r',
            'Dark2', 'Dark2_r', 'GnBu', 'GnBu_r', 'Greens', 'Greens_r',
            'Greys', 'Greys_r', 'OrRd', 'OrRd_r', 'Oranges', 'Oranges_r',
            'PRGn', 'PRGn_r', 'Paired', 'Paired_r', 'Pastel1', 'Pastel1_r',
            'Pastel2', 'Pastel2_r', 'PiYG', 'PiYG_r', 'PuBu', 'PuBuGn',
            'PuBuGn_r', 'PuBu_r', 'PuOr', 'PuOr_r', 'PuRd', 'PuRd_r',
            'Purples', 'Purples_r', 'RdBu', 'RdBu_r', 'RdGy', 'RdGy_r',
            'RdPu', 'RdPu_r', 'RdYlBu', 'RdYlBu_r', 'RdYlGn', 'RdYlGn_r',
            'Reds', 'Reds_r', 'Set1', 'Set1_r', 'Set2', 'Set2_r', 'Set3',
            'Set3_r', 'Spectral', 'Spectral_r',
            #'Vega10', 'Vega10_r', 'Vega20',
            'Vega20_r', 'Vega20b', 'Vega20b_r', 'Vega20c', 'Vega20c_r',
            'Wistia', 'Wistia_r', 'YlGn', 'YlGnBu', 'YlGnBu_r', 'YlGn_r',
            'YlOrBr', 'YlOrBr_r', 'YlOrRd', 'YlOrRd_r', 'afmhot',
'afmhot_r',
```

```
            'autumn', 'autumn_r', 'binary', 'binary_r', 'bone', 'bone_r',
            'brg', 'brg_r', 'bwr', 'bwr_r', 'cool', 'cool_r', 'coolwarm',
            'coolwarm_r', 'copper', 'copper_r', 'cubehelix', 'cubehelix_r',
            'flag', 'flag_r', 'gist_earth', 'gist_earth_r', 'gist_gray',
            'gist_gray_r', 'gist_heat', 'gist_heat_r', 'gist_ncar',
            'gist_ncar_r', 'gist_rainbow', 'gist_rainbow_r', 'gist_stern',
            'gist_stern_r', 'gist_yarg', 'gist_yarg_r', 'gnuplot',
'gnuplot2',
            'gnuplot2_r', 'gnuplot_r', 'gray', 'gray_r', 'hot', 'hot_r',
'hsv',
            'hsv_r', 'icefire', 'icefire_r', 'inferno', 'inferno_r',
            #'jet', ##'jet_r',
            'magma', 'magma_r', 'mako', 'mako_r', 'nipy_spectral',
            'nipy_spectral_r', 'ocean', 'ocean_r', 'pink', 'pink_r',
'plasma',
            'plasma_r', 'prism', 'prism_r', 'rainbow', 'rainbow_r',
'rocket',
            'rocket_r', 'seismic', 'seismic_r', ##'spectral', 'spectral_r',
            'spring', 'spring_r', 'summer', 'summer_r', 'tab10', 'tab10_r',
            'tab20', 'tab20_r', 'tab20b', 'tab20b_r', 'tab20c', 'tab20c_r',
            'terrain', 'terrain_r', 'viridis', 'viridis_r', 'vlag',
'vlag_r',
            'winter', 'winter_r']

# Definições de estilo
if 1==1: #tema Claro para artigo retrato
    DefPalette = palettes[13] #13 Dark2
    DefPaletteHist = 'rocket'
    DefStyle = stl[15] #15 Seaborn-Darkgrid
    DefSize = (10,6) #(17,6)
    DefGrid = ('-')

else: # tema Escuro para apresentação paisagem
    DefPalette = palettes[123] #123 - hsv
    DefPaletteHist = 'rocket'
    DefStyle = stl[4] #15 dark_background
    DefSize = (10,6) #(17,6)
    DefGrid = (':')
    plt.rc_context({'axes.edgecolor':'gray', 'xtick.color':'white',
                    'ytick.color':'white', 'figure.facecolor':'black'})



ibov = pd.read_csv(urlibov, sep=';', decimal=',', thousands='.',
skiprows=1, skip_blank_lines=True, header=0, encoding='Latin-1')
ibov['mm'] = '0' + ibov['Mês'].astype(str)
ibov['Data'] = ibov.Ano.astype(str) + '-' +ibov.mm.str[-2:]+ '-01'
ibov['Data'] = ibov['Data'].astype("datetime64")
ibov.rename(columns={'Valor':'ibov'}, inplace = True)
ibov = ibov.reset_index().set_index('Data')
#ibov = ibov['ibov']
ibov.head(2)

ibov2013_2021 = ibov[(ibov.index.year >= 2013) & (ibov.index.year <= 2021)]

ibov2013_2021

plt.figure(figsize=(16,10))
sns.set_theme()
sns.lineplot(data = ibov2013_2021, x= ibov2013_2021.index.month, y='ibov',
hue=ibov2013_2021.index.year)

ibov.index
```

```
plt.figure(figsize=(16,10))
sns.set_theme()
sns.lineplot(data = ibov.reset_index()[-60:], x= ibov.index[-60:],
y='ibov')#, hue=ibcbr.index.year)

decomposicao = seasonal_decompose(ibov2013_2021.ibov)
decomposicao.plot()
plt.gcf().set_size_inches(DefSize)
plt.show();

decomposicao = seasonal_decompose(ibov2013_2021.ibov,
model='multiplicative')
sns.set_theme()
style.use(DefStyle)
ax = decomposicao.plot()
plt.gcf().set_size_inches(DefSize)

plt.show();

ax = plot_acf(ibov.ibov,lags=60);
style.use(DefStyle)
#plt.rc_context({'axes.edgecolor':'gray', 'xtick.color':'white',
'ytick.color':'white', 'figure.facecolor':'black'})

plt.title('Teste de Auto Correlação')
plt.gcf().set_size_inches(10, 6)
plt.grid(True,linestyle = DefGrid)
plt.show();
```

IBCBR

```
"""ibcBR_2013_2021.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/github/lvb86/PD_LSTM_GA/blob/main/code/ibcBR_2013_2021.ipynb

# Etapa de análise e limpeza de dados

Preparando dados IBC-BR
por: Leandro Barbosa
"""

# Commented out IPython magic to ensure Python compatibility.
import math
import matplotlib.pyplot    as plt
import matplotlib.ticker    as ticker
import matplotlib.style     as style
import numpy                as np
import seaborn              as sns
import pandas               as pd

from urllib.request                 import urlopen
from zipfile                        import ZipFile
from statsmodels.graphics.tsaplots  import plot_acf, plot_pacf
from statsmodels.tsa.seasonal       import seasonal_decompose
from sklearn.metrics                import mean_squared_error
#from sklearn.metrics                import mean_absolute_percentage_error
from sklearn.linear_model           import LinearRegression

stl = style.available
# %matplotlib inline

prefGo = 'https://docs.google.com/uc?export=download&id='
urlibc = prefGo + '1QrbgeR7TyHbcNx3l-ke2RD33kn0p0BgY'

mes =
['Jan','Fev','Mar','Abr','Mai','Jun','Jul','Ago','Set','Out','Nov','Dez']
mesN = [1,2,3,4,5,6,7,8,9,10,11,12]

palettes = ['Accent', 'Accent_r', 'Blues', 'Blues_r', 'BrBG', 'BrBG_r',
            'BuGn', 'BuGn_r', 'BuPu', 'BuPu_r', 'CMRmap', 'CMRmap_r',
            'Dark2', 'Dark2_r', 'GnBu', 'GnBu_r', 'Greens', 'Greens_r',
            'Greys', 'Greys_r', 'OrRd', 'OrRd_r', 'Oranges', 'Oranges_r',
            'PRGn', 'PRGn_r', 'Paired', 'Paired_r', 'Pastel1', 'Pastel1_r',
            'Pastel2', 'Pastel2_r', 'PiYG', 'PiYG_r', 'PuBu', 'PuBuGn',
            'PuBuGn_r', 'PuBu_r', 'PuOr', 'PuOr_r', 'PuRd', 'PuRd_r',
            'Purples', 'Purples_r', 'RdBu', 'RdBu_r', 'RdGy', 'RdGy_r',
            'RdPu', 'RdPu_r', 'RdYlBu', 'RdYlBu_r', 'RdYlGn', 'RdYlGn_r',
            'Reds', 'Reds_r', 'Set1', 'Set1_r', 'Set2', 'Set2_r', 'Set3',
            'Set3_r', 'Spectral', 'Spectral_r',
            #'Vega10', 'Vega10_r', 'Vega20',
            'Vega20_r', 'Vega20b', 'Vega20b_r', 'Vega20c', 'Vega20c_r',
            'Wistia', 'Wistia_r', 'YlGn', 'YlGnBu', 'YlGnBu_r', 'YlGn_r',
            'YlOrBr', 'YlOrBr_r', 'YlOrRd', 'YlOrRd_r', 'afmhot',
'afmhot_r',
            'autumn', 'autumn_r', 'binary', 'binary_r', 'bone', 'bone_r',
            'brg', 'brg_r', 'bwr', 'bwr_r', 'cool', 'cool_r', 'coolwarm',
            'coolwarm_r', 'copper', 'copper_r', 'cubehelix', 'cubehelix_r',
```

```
            'flag', 'flag_r', 'gist_earth', 'gist_earth_r', 'gist_gray',
            'gist_gray_r', 'gist_heat', 'gist_heat_r', 'gist_ncar',
            'gist_ncar_r', 'gist_rainbow', 'gist_rainbow_r', 'gist_stern',
            'gist_stern_r', 'gist_yarg', 'gist_yarg_r', 'gnuplot',
'gnuplot2',
            'gnuplot2_r', 'gnuplot_r', 'gray', 'gray_r', 'hot', 'hot_r',
'hsv',
            'hsv_r', 'icefire', 'icefire_r', 'inferno', 'inferno_r',
            #'jet', ##'jet_r',
            'magma', 'magma_r', 'mako', 'mako_r', 'nipy_spectral',
            'nipy_spectral_r', 'ocean', 'ocean_r', 'pink', 'pink_r',
'plasma',
            'plasma_r', 'prism', 'prism_r', 'rainbow', 'rainbow_r',
'rocket',
            'rocket_r', 'seismic', 'seismic_r', ##'spectral', 'spectral_r',
            'spring', 'spring_r', 'summer', 'summer_r', 'tab10', 'tab10_r',
            'tab20', 'tab20_r', 'tab20b', 'tab20b_r', 'tab20c', 'tab20c_r',
            'terrain', 'terrain_r', 'viridis', 'viridis_r', 'vlag',
'vlag_r',
            'winter', 'winter_r']

# Definições de estilo
if 1==1: #tema Claro para artigo retrato
    DefPalette = palettes[13] #13 Dark2
    DefPaletteHist = 'rocket'
    DefStyle = stl[15] #15 Seaborn-Darkgrid
    DefSize = (10,6) #(17,6)
    DefGrid = ('-')

else: # tema Escuro para apresentação paisagem
    DefPalette = palettes[123] #123 - hsv
    DefPaletteHist = 'rocket'
    DefStyle = stl[4] #15 dark_background
    DefSize = (10,6) #(17,6)
    DefGrid = (':')
    plt.rc_context({'axes.edgecolor':'gray', 'xtick.color':'white',
                    'ytick.color':'white', 'figure.facecolor':'black'})

ibcbr = pd.read_csv(urlibc, sep =';' ,decimal=',',parse_dates=True,
dayfirst=True,encoding='UTF8', index_col='data' )

ibcbr.head(12).index.month

ibcbr2013_2021 = ibcbr[(ibcbr.index.year >= 2013) & (ibcbr.index.year <=
2021)]

ibcbr2013_2021.to_csv('/tmp/ibcbr2013_2021.csv',index= False)

plt.figure(figsize=(16,10))
sns.lineplot(data = ibcbr2013_2021, x= ibcbr2013_2021.index.month,
y='valor', hue=ibcbr2013_2021.index.year)

plt.figure(figsize=(16,10))
sns.set_theme()
sns.lineplot(data = ibcbr[-24:], x= 'data', y='valor')#,
hue=ibcbr.index.year)

plt.figure(figsize=(16,10))
sns.set_theme()
sns.lineplot(data = ibcbr2013_2021[0:12], x=
ibcbr2013_2021.index.month[0:12], y='valor')#,
hue=ibcbr2013_2018.index.year)
```

```
ibcbr2013_2021

ibcbr2016_2021 = ibcbr[(ibcbr.index.year >= 2016) & (ibcbr.index.year <=
2021)]

plt.figure(figsize=(16,10))
sns.lineplot(data = ibcbr2016_2021, x= ibcbr2016_2021.index.month,
y='valor', hue=ibcbr2016_2021.index.year)

ibcbr2016_2021

plt.figure(figsize=(16,10))
sns.set_theme()
sns.lineplot(ibcbr2016_2021.reset_index().data,ibcbr2016_2021.reset_index()
.valor, palette=DefPalette)

decomposicao = seasonal_decompose(ibcbr2013_2021)
decomposicao.plot()
plt.gcf().set_size_inches(DefSize)
plt.show();

decomposicao = seasonal_decompose(ibcbr2013_2021, model='multiplicative')
sns.set_theme()
style.use(DefStyle)
ax = decomposicao.plot()
plt.gcf().set_size_inches(DefSize)

plt.show();

ax = plot_acf(ibcbr,lags=60);
style.use(DefStyle)
#plt.rc_context({'axes.edgecolor':'gray', 'xtick.color':'white',
'ytick.color':'white', 'figure.facecolor':'black'})

plt.title('Teste de Auto Correlação')
plt.gcf().set_size_inches(10, 6)
plt.grid(True,linestyle = DefGrid)
plt.show();
```

IPCA

```
"""IPCA.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/github/lvb86/PD_LSTM_GA/blob/main/code/IP
CA.ipynb

# Etapa de análise e limpeza de dados

Preparando dados IBC-BR
por: Leandro Barbosa
"""

# Commented out IPython magic to ensure Python compatibility.
import math
import matplotlib.pyplot    as plt
import matplotlib.ticker    as ticker
import matplotlib.style     as style
import numpy                as np
import seaborn              as sns
import pandas               as pd

from urllib.request                 import urlopen
from zipfile                        import ZipFile
from statsmodels.graphics.tsaplots  import plot_acf, plot_pacf
from statsmodels.tsa.seasonal       import seasonal_decompose
from sklearn.metrics                import mean_squared_error
#from sklearn.metrics                import mean_absolute_percentage_error
from sklearn.linear_model           import LinearRegression

stl = style.available
# %matplotlib inline

prefGo  = 'https://docs.google.com/uc?export=download&id='
urlipca = prefGo + '1l6wRFAprymsQoVoNqAvv1g2EMN2tmVMa'

mes =
['Jan','Fev','Mar','Abr','Mai','Jun','Jul','Ago','Set','Out','Nov','Dez']
mesN = [1,2,3,4,5,6,7,8,9,10,11,12]

palettes = ['Accent', 'Accent_r', 'Blues', 'Blues_r', 'BrBG', 'BrBG_r',
            'BuGn', 'BuGn_r', 'BuPu', 'BuPu_r', 'CMRmap', 'CMRmap_r',
            'Dark2', 'Dark2_r', 'GnBu', 'GnBu_r', 'Greens', 'Greens_r',
            'Greys', 'Greys_r', 'OrRd', 'OrRd_r', 'Oranges', 'Oranges_r',
            'PRGn', 'PRGn_r', 'Paired', 'Paired_r', 'Pastel1', 'Pastel1_r',
            'Pastel2', 'Pastel2_r', 'PiYG', 'PiYG_r', 'PuBu', 'PuBuGn',
            'PuBuGn_r', 'PuBu_r', 'PuOr', 'PuOr_r', 'PuRd', 'PuRd_r',
            'Purples', 'Purples_r', 'RdBu', 'RdBu_r', 'RdGy', 'RdGy_r',
            'RdPu', 'RdPu_r', 'RdYlBu', 'RdYlBu_r', 'RdYlGn', 'RdYlGn_r',
            'Reds', 'Reds_r', 'Set1', 'Set1_r', 'Set2', 'Set2_r', 'Set3',
            'Set3_r', 'Spectral', 'Spectral_r',
            #'Vega10', 'Vega10_r', 'Vega20',
            'Vega20_r', 'Vega20b', 'Vega20b_r', 'Vega20c', 'Vega20c_r',
            'Wistia', 'Wistia_r', 'YlGn', 'YlGnBu', 'YlGnBu_r', 'YlGn_r',
            'YlOrBr', 'YlOrBr_r', 'YlOrRd', 'YlOrRd_r', 'afmhot',
'afmhot_r',
            'autumn', 'autumn_r', 'binary', 'binary_r', 'bone', 'bone_r',
            'brg', 'brg_r', 'bwr', 'bwr_r', 'cool', 'cool_r', 'coolwarm',
            'coolwarm_r', 'copper', 'copper_r', 'cubehelix', 'cubehelix_r',
```

```
            'flag', 'flag_r', 'gist_earth', 'gist_earth_r', 'gist_gray',
            'gist_gray_r', 'gist_heat', 'gist_heat_r', 'gist_ncar',
            'gist_ncar_r', 'gist_rainbow', 'gist_rainbow_r', 'gist_stern',
            'gist_stern_r', 'gist_yarg', 'gist_yarg_r', 'gnuplot',
'gnuplot2',
            'gnuplot2_r', 'gnuplot_r', 'gray', 'gray_r', 'hot', 'hot_r',
'hsv',
            'hsv_r', 'icefire', 'icefire_r', 'inferno', 'inferno_r',
            #'jet', ##'jet_r',
            'magma', 'magma_r', 'mako', 'mako_r', 'nipy_spectral',
            'nipy_spectral_r', 'ocean', 'ocean_r', 'pink', 'pink_r',
'plasma',
            'plasma_r', 'prism', 'prism_r', 'rainbow', 'rainbow_r',
'rocket',
            'rocket_r', 'seismic', 'seismic_r', ##'spectral', 'spectral_r',
            'spring', 'spring_r', 'summer', 'summer_r', 'tab10', 'tab10_r',
            'tab20', 'tab20_r', 'tab20b', 'tab20b_r', 'tab20c', 'tab20c_r',
            'terrain', 'terrain_r', 'viridis', 'viridis_r', 'vlag',
'vlag_r',
            'winter', 'winter_r']

# Definições de estilo
if 1==1: #tema Claro para artigo retrato
    DefPalette = palettes[13] #13 Dark2
    DefPaletteHist = 'rocket'
    DefStyle = stl[15] #15 Seaborn-Darkgrid
    DefSize = (10,6) #(17,6)
    DefGrid = ('-')

else: # tema Escuro para apresentação paisagem
    DefPalette = palettes[123] #123 - hsv
    DefPaletteHist = 'rocket'
    DefStyle = stl[4] #15 dark_background
    DefSize = (10,6) #(17,6)
    DefGrid = (':')
    plt.rc_context({'axes.edgecolor':'gray', 'xtick.color':'white',
                    'ytick.color':'white', 'figure.facecolor':'black'})

ipca

ipca = pd.read_csv(urlipca, sep =';' ,decimal=',',parse_dates=True,
dayfirst=True,encoding='UTF8', index_col='data' )

ipca.rename(columns={'valor':'ipca'}, inplace = True)
ipca.head(2)

ipca2013_2021 = ipca[(ipca.index.year >= 2013) & (ipca.index.year <= 2021)]

ipca2013_2021.head()

plt.figure(figsize=(16,10))
sns.set_theme()
sns.lineplot(data = ipca2013_2021, x= ipca2013_2021.index.month, y='ipca',
hue=ipca2013_2021.index.year)

ipca.index

plt.figure(figsize=(16,10))
sns.set_theme()
sns.lineplot(data = ipca.reset_index()[-60:], x= ipca.index[-60:],
y='ipca')#, hue=ibcbr.index.year)

decomposicao = seasonal_decompose(ipca2013_2021.ipca)
```

```
decomposicao.plot()
plt.gcf().set_size_inches(DefSize)
plt.show();

decomposicao = seasonal_decompose(ipca2013_2021.ipca,
model='multiplicative')
sns.set_theme()
style.use(DefStyle)
ax = decomposicao.plot()
plt.gcf().set_size_inches(DefSize)

plt.show();

ax = plot_acf(ipca.ipca,lags=60);
style.use(DefStyle)
#plt.rc_context({'axes.edgecolor':'gray', 'xtick.color':'white',
'ytick.color':'white', 'figure.facecolor':'black'})

plt.title('Teste de Auto Correlação')
plt.gcf().set_size_inches(10, 6)
plt.grid(True,linestyle = DefGrid)
plt.show();
```

Óbitos

```
"""Obitos_Tabnet_2013_2021.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/github/lvb86/PD_LSTM_GA/blob/main/code/Obitos_Tabnet_2013_2021.ipynb
"""

pip install -U scikit-learn

# Commented out IPython magic to ensure Python compatibility.
import math
import matplotlib.pyplot    as plt
import matplotlib.ticker    as ticker
import matplotlib.style     as style
import numpy                as np
import seaborn              as sns
import pandas               as pd

from urllib.request                 import urlopen
from zipfile                        import ZipFile
from statsmodels.graphics.tsaplots  import plot_acf, plot_pacf
from statsmodels.tsa.seasonal       import seasonal_decompose
from sklearn.metrics                import mean_squared_error
from sklearn.metrics                import mean_absolute_percentage_error
from sklearn.linear_model           import LinearRegression

stl = style.available
# %matplotlib inline

prefGo    = 'https://docs.google.com/uc?export=download&id='
urlTabnet = prefGo + '1z1vn0D9Efnl-wRO_tYDZafZhLGRrPwc0'

mes =
['Jan','Fev','Mar','Abr','Mai','Jun','Jul','Ago','Set','Out','Nov','Dez']
mesN = [1,2,3,4,5,6,7,8,9,10,11,12]

palettes = ['Accent', 'Accent_r', 'Blues', 'Blues_r', 'BrBG', 'BrBG_r',
            'BuGn', 'BuGn_r', 'BuPu', 'BuPu_r', 'CMRmap', 'CMRmap_r',
            'Dark2', 'Dark2_r', 'GnBu', 'GnBu_r', 'Greens', 'Greens_r',
            'Greys', 'Greys_r', 'OrRd', 'OrRd_r', 'Oranges', 'Oranges_r',
            'PRGn', 'PRGn_r', 'Paired', 'Paired_r', 'Pastel1', 'Pastel1_r',
            'Pastel2', 'Pastel2_r', 'PiYG', 'PiYG_r', 'PuBu', 'PuBuGn',
            'PuBuGn_r', 'PuBu_r', 'PuOr', 'PuOr_r', 'PuRd', 'PuRd_r',
            'Purples', 'Purples_r', 'RdBu', 'RdBu_r', 'RdGy', 'RdGy_r',
            'RdPu', 'RdPu_r', 'RdYlBu', 'RdYlBu_r', 'RdYlGn', 'RdYlGn_r',
            'Reds', 'Reds_r', 'Set1', 'Set1_r', 'Set2', 'Set2_r', 'Set3',
            'Set3_r', 'Spectral', 'Spectral_r',
            #'Vega10', 'Vega10_r', 'Vega20',
            'Vega20_r', 'Vega20b', 'Vega20b_r', 'Vega20c', 'Vega20c_r',
            'Wistia', 'Wistia_r', 'YlGn', 'YlGnBu', 'YlGnBu_r', 'YlGn_r',
            'YlOrBr', 'YlOrBr_r', 'YlOrRd', 'YlOrRd_r', 'afmhot',
'afmhot_r',
            'autumn', 'autumn_r', 'binary', 'binary_r', 'bone', 'bone_r',
            'brg', 'brg_r', 'bwr', 'bwr_r', 'cool', 'cool_r', 'coolwarm',
            'coolwarm_r', 'copper', 'copper_r', 'cubehelix', 'cubehelix_r',
            'flag', 'flag_r', 'gist_earth', 'gist_earth_r', 'gist_gray',
            'gist_gray_r', 'gist_heat', 'gist_heat_r', 'gist_ncar',
            'gist_ncar_r', 'gist_rainbow', 'gist_rainbow_r', 'gist_stern',
```

```
            'gist_stern_r', 'gist_yarg', 'gist_yarg_r', 'gnuplot',
'gnuplot2',
            'gnuplot2_r', 'gnuplot_r', 'gray', 'gray_r', 'hot', 'hot_r',
'hsv',
            'hsv_r', 'icefire', 'icefire_r', 'inferno', 'inferno_r',
            #'jet', ##'jet_r',
            'magma', 'magma_r', 'mako', 'mako_r', 'nipy_spectral',
            'nipy_spectral_r', 'ocean', 'ocean_r', 'pink', 'pink_r',
'plasma',
            'plasma_r', 'prism', 'prism_r', 'rainbow', 'rainbow_r',
'rocket',
            'rocket_r', 'seismic', 'seismic_r', ##'spectral', 'spectral_r',
            'spring', 'spring_r', 'summer', 'summer_r', 'tab10', 'tab10_r',
            'tab20', 'tab20_r', 'tab20b', 'tab20b_r', 'tab20c', 'tab20c_r',
            'terrain', 'terrain_r', 'viridis', 'viridis_r', 'vlag',
'vlag_r',
            'winter', 'winter_r']

# Definições de estilo
if 1==1: #tema Claro para artigo retrato
    DefPalette = palettes[13] #13 Dark2
    DefPaletteHist = 'rocket'
    DefStyle = stl[15] #15 Seaborn-Darkgrid
    DefSize = (10,6) #(17,6)
    DefGrid = ('-')

else: # tema Escuro para apresentação paisagem
    DefPalette = palettes[123] #123 - hsv
    DefPaletteHist = 'rocket'
    DefStyle = stl[4] #15 dark_background
    DefSize = (10,6) #(17,6)
    DefGrid = (':')
    plt.rc_context({'axes.edgecolor':'gray', 'xtick.color':'white',
                    'ytick.color':'white', 'figure.facecolor':'black'})

tabnet = pd.read_csv(urlTabnet, sep =';', decimal=',', skiprows=3,
skipfooter=6, encoding='latin_1')

tabnet.head()

tabnet['Ano/mês processamento'] = tabnet['Ano/mês
processamento'].str.replace('..','',regex=False)
tabnet = tabnet[tabnet['Ano/mês processamento'].str.len() >= 6]
tabnet['Ano'] = tabnet['Ano/mês processamento'].str[-4:]
tabnet['Mês'] = tabnet['Ano/mês processamento'].str[:-5]
tabnet.drop(columns = 'Ano/mês processamento')

mapmes = {'Janeiro':1, 'Fevereiro':2, 'Março':3, 'Abril':4, 'Maio':5,
'Junho':6,
 'Julho':7, 'Agosto':8, 'Setembro':9, 'Outubro':10, 'Novembro':11,
'Dezembro':12}

tabnet['Mês'] = tabnet['Mês'].str.strip()
tabnet['mm'] = tabnet['Mês'].map(mapmes)

tabnet['Ano_mes'] = tabnet.Ano.astype(str) + '-' +
('0'+tabnet.mm.astype(str).str[-2:])

tabnet.mm.unique()

plt.figure(figsize = (16,10))
sns.lineplot(data=tabnet, x = tabnet.index, y=tabnet.Total)
```

```
plt.figure(figsize = (8,6))
ax = sns.lineplot(data=tabnet, x = tabnet.Ano_mes, y=tabnet.Total)
ax.set_xticks(ax.get_xticks()[::12])
plt.xticks(rotation = 45)
plt.xlabel('Ano Mês')
plt.ylabel('Total de Óbitos')
plt.show();

plt.figure(figsize = (16,10))
sns.lineplot(data=tabnet, x = tabnet.mm, y=tabnet.Total, hue=tabnet.Ano)

tabnet = pd.read_csv(urlTabnet, sep =';', decimal=',', skiprows=3,
skipfooter=6, encoding='latin_1')
tabnet['Ano/mês processamento'] = tabnet['Ano/mês
processamento'].str.replace('..','',regex=False)
tabnet = tabnet[tabnet['Ano/mês processamento'].str.len() >= 6]
tabnet['Ano'] = tabnet['Ano/mês processamento'].str[-4:]
tabnet['Mês'] = tabnet['Ano/mês processamento'].str[:-5]
tabnet['Mês'] = tabnet['Mês'].str.strip()
mapmes = {'Janeiro':'01', 'Fevereiro':'02', 'Março':'03', 'Abril':'04',
'Maio':'05', 'Junho':'06',
 'Julho':'07', 'Agosto':'08', 'Setembro':'09', 'Outubro':'10',
'Novembro':'11', 'Dezembro':'12'}
tabnet['mm'] = tabnet['Mês'].map(mapmes)
tabnet.drop(columns = 'Ano/mês processamento')
tabnet['Ano Mês'] = tabnet.Ano +'-'+ tabnet.mm
tabnet = tabnet.rename(columns = {'Total': 'BR'})
tabnet
```

Testes de Hiper Parâmetros:

```
"""Testes_de_Hiper_Parâmetros.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/github/lvb86/PD_LSTM_GA/blob/main/code/Testes_de_Hiper_Par%C3%A2metros.ipynb

#Pesquisa de Hiperparâmetros
Na MLP da Biblioteca Scikit-learn

#Declarações Globais
"""

pip install -U scikit-learn

import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
from sklearn.model_selection    import
train_test_split,RepeatedStratifiedKFold
from sklearn.preprocessing      import StandardScaler         #Normalização
dos dados
from sklearn.decomposition      import PCA                    #Redução de
Dimencionalidade
from sklearn.pipeline           import make_pipeline          #Pipe Line
from sklearn.neural_network     import MLPRegressor
from sklearn.metrics            import mean_absolute_percentage_error

from sklearn.model_selection import GridSearchCV
#from sklearn.svm import SVC
seed = 170696
np.seed = seed

urldfCC =
'https://raw.githubusercontent.com/lvb86/MLPr_GA/main/swap/dfConsumoClima2.
csv'
df = pd.read_csv(urldfCC)

d = df.iloc[:,2:-3]
d.head()

"""Objetivo prever o ano seguinte ...
como a base é 2013-2019

treino com 2013- 2017
teste com 2018
"""

X_train = d.iloc[:-12,].drop('Consumo', axis =1)
y_train = d['Consumo'].iloc[:-12,]

X_test = d.iloc[-12:,].drop('Consumo', axis =1)
y_test = d['Consumo'].iloc[-12:,]

"""Funções Compartilhadas"""

def graf_bar(x,y,Xlabel, rot):
    sns.set_style('darkgrid')
```

```
    plt.figure(figsize=(12,8))
    g= sns.barplot(x=x,y=y, palette='rocket')

    for i in range(len(x)):
        g.text(i,y[i]+0.001, round(y[i],3), color='black', ha="center",
fontsize=15)
    plt.xticks(rotation = rot, fontsize=15)
    #plt.ylim(0,1.1)
    plt.ylabel('MAPE')
    plt.xlabel(Xlabel)

    plt.show();

"""#Testes

## Melhor configuração
"""

#hidden_layer_sizes = (8,16,32,4)
hidden_layer_sizes = (2,3,59,38)
activation='relu'
random_state=1
max_iter=100_000
alpha = 0.0001
learning_rate_init = 0.001
shuffle = True
beta_1 = 0.94
beta_2 = 0.999
epsilon = 0.000_000_000_1
n_iter_no_change=1000
solver='adam'
verbose = False

pipe_mlp = make_pipeline(StandardScaler(),
                         #PCA(n_components=i),
                         MLPRegressor(hidden_layer_sizes=
hidden_layer_sizes,
                                      activation=activation,
                                      random_state=random_state,
                                      max_iter=max_iter,
                                      alpha = alpha,
                                      learning_rate_init =
learning_rate_init,
                                      shuffle = shuffle,
                                      beta_1 = beta_1,
                                      beta_2 = beta_2,
                                      epsilon = epsilon,
                                      n_iter_no_change=n_iter_no_change,
                                      solver=solver,
                                      verbose = verbose
                                     )
                         )

pipe_mlp.get_params()

"""## Estrutura"""

dic_param = {
1:(2,),
2:(8, 4),
3:(8, 4, 2),
4:(8, 16, 4, 2),
5:(8, 16, 8, 4, 2),
```

```
6:(8, 16, 32, 8, 4, 2),
7:(8, 16, 32, 16, 8, 4, 2),
8:(8, 16, 32, 64, 16, 8, 4, 2),
9:(8, 16, 32, 64, 32, 16, 8, 4, 2),
10:(8, 16, 32, 64, 128, 32, 16, 8, 4, 2)}

dic_param2 = {
1:(8,),
2:(8, 4),
3:(8, 8, 4),
4:(8, 16, 32, 4),
5:(8, 16, 32, 64, 4),
6:(8, 16, 32, 64, 128, 4),
7:(8, 16, 32, 64, 128, 256, 4),
8:(8, 16, 32, 64, 128, 256, 512, 4),
9:(8, 16, 32, 64, 128, 256, 512, 1024, 4),
10:(8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4)}

dic_param3 = {
1:(8,16,32,2),
2:(8,32,16,2),
3:(8,32,16,4),
4:(8,32,64,2),
5:(8,64,32,2),
6:(8,32,64,4),
7:(8,64,32,4)}

mape = []
Lparam = []
for i in dic_param3:
    print(dic_param3[i])
    p = dic_param3[i]
    pipe_mlp = make_pipeline(StandardScaler(),
                             #PCA(n_components=i),
                             MLPRegressor(hidden_layer_sizes= p,
                                          activation=activation,
                                          random_state=random_state,
                                          max_iter=max_iter,
                                          alpha = alpha,
                                          learning_rate_init =
learning_rate_init,
                                          shuffle = shuffle,
                                          beta_1 = beta_1,
                                          beta_2 = beta_2,
                                          epsilon = epsilon,

n_iter_no_change=n_iter_no_change,
                                          solver=solver,
                                          verbose = verbose
                                      )
                             )

    pipe_mlp.fit(X_train, y_train)
    y_pred = pipe_mlp.predict(X_test)
    m = mean_absolute_percentage_error(y_test,y_pred)
    mape.append(m)
    Lparam.append(str(p))
    print('mape','|',m)
graf_bar(Lparam,mape,'hidden_layer_sizes', 30)

"""resultados agrupados"""
```

```
url =
'https://github.com/lvb86/MLPr_GA/blob/main/swap/ResultadosRede.xlsx?raw=tr
ue'
top10=pd.read_excel(url, ).iloc[0:10,:]

top10

graf_bar(top10.Estrutura,top10.MAPE,'Estrutura',90)

"""##Função de ativação"""

param = ['relu','identity','tanh','logistic']
mape = []
for e, p in enumerate(param):
    pipe_mlp = make_pipeline(StandardScaler(),
                             #PCA(n_components=i),
                             MLPRegressor(hidden_layer_sizes=
hidden_layer_sizes,
                                          activation=p,
                                          random_state=random_state,
                                          max_iter=max_iter,
                                          alpha = alpha,
                                          learning_rate_init =
learning_rate_init,
                                          shuffle = shuffle,
                                          beta_1 = beta_1,
                                          beta_2 = beta_2,
                                          epsilon = epsilon,

n_iter_no_change=n_iter_no_change,
                                          solver=solver,
                                          verbose = verbose
                                      )
                             )

    pipe_mlp.fit(X_train, y_train)
    y_pred = pipe_mlp.predict(X_test)
    m = mean_absolute_percentage_error(y_test,y_pred)
    mape.append(m)
    print('mape','|',m)
graf_bar(param,mape,'Função de ativação',0)

"""##Solver"""

param = ['lbfgs', 'sgd', 'adam']

mape = []
for e, p in enumerate(param):
    pipe_mlp = make_pipeline(StandardScaler(),
                             #PCA(n_components=i),
                             MLPRegressor(hidden_layer_sizes=
hidden_layer_sizes,
                                          activation=activation,
                                          random_state=random_state,
                                          max_iter=max_iter,
                                          alpha = alpha,
                                          learning_rate_init =
learning_rate_init,
                                          shuffle = shuffle,
                                          beta_1 = beta_1,
                                          beta_2 = beta_2,
                                          epsilon = epsilon,
```

```
n_iter_no_change=n_iter_no_change,
                                                    solver=p,
                                                    verbose = verbose
                                                )
                                )

    pipe_mlp.fit(X_train, y_train)
    y_pred = pipe_mlp.predict(X_test)
    m = mean_absolute_percentage_error(y_test,y_pred)
    mape.append(m)
    print('mape','|',m)
graf_bar(param,mape,'solver', 0)

"""##Alpha"""

param = [1,0.1,0.01,0.001,0.0001,0.00001]
Lparam = []
mape = []
for e, p in enumerate(param):
    pipe_mlp = make_pipeline(StandardScaler(),
                             #PCA(n_components=i),
                             MLPRegressor(hidden_layer_sizes=
hidden_layer_sizes,
                                                    activation=activation,
                                                    random_state=random_state,
                                                    max_iter=max_iter,
                                                    alpha = p,
                                                    learning_rate_init =
learning_rate_init,
                                                    shuffle = shuffle,
                                                    beta_1 = beta_1,
                                                    beta_2 = beta_2,
                                                    epsilon = epsilon,

n_iter_no_change=n_iter_no_change,
                                                    solver=solver,
                                                    verbose = verbose
                                                )
                                )

    pipe_mlp.fit(X_train, y_train)
    y_pred = pipe_mlp.predict(X_test)
    m = mean_absolute_percentage_error(y_test,y_pred)
    mape.append(m)
    Lparam.append(str(p))
    print('mape','|',m)
graf_bar(Lparam,mape,'alpha', 0)

"""##Learning Rate"""

param = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0, 2.0]
Lparam = []
mape = []
for e, p in enumerate(param):
    pipe_mlp = make_pipeline(StandardScaler(),
                             #PCA(n_components=i),
                             MLPRegressor(hidden_layer_sizes=
hidden_layer_sizes,
                                                    activation=activation,
                                                    random_state=random_state,
                                                    max_iter=max_iter,
                                                    alpha = alpha,
```

```
                                                learning_rate_init = p,
                                                shuffle = shuffle,
                                                beta_1 = beta_1,
                                                beta_2 = beta_2,
                                                epsilon = epsilon,

n_iter_no_change=n_iter_no_change,
                                                solver=solver,
                                                verbose = verbose
                                            )
                                )

    pipe_mlp.fit(X_train, y_train)
    y_pred = pipe_mlp.predict(X_test)
    m = mean_absolute_percentage_error(y_test,y_pred)
    mape.append(m)
    Lparam.append(str(p))
    print('mape','|',m)
graf_bar(Lparam,mape,'learning_rate_init', 30)

"""##Shuffle"""

param = [True, False]
Lparam = []
mape = []
for e, p in enumerate(param):
    pipe_mlp = make_pipeline(StandardScaler(),
                                #PCA(n_components=i),
                                MLPRegressor(hidden_layer_sizes=
hidden_layer_sizes,
                                                activation=activation,
                                                random_state=random_state,
                                                max_iter=max_iter,
                                                alpha = alpha,
                                                learning_rate_init =
learning_rate_init,
                                                shuffle = p,
                                                beta_1 = beta_1,
                                                beta_2 = beta_2,
                                                epsilon = epsilon,

n_iter_no_change=n_iter_no_change,
                                                solver=solver,
                                                verbose = verbose
                                            )
                                )

    pipe_mlp.fit(X_train, y_train)
    y_pred = pipe_mlp.predict(X_test)
    m = mean_absolute_percentage_error(y_test,y_pred)
    mape.append(m)
    Lparam.append(str(p))
    print('mape','|',m)
graf_bar(Lparam,mape,'suffle', 0)

"""##Beta 1 """

param = [0.001, 0.01, 0.1, 0.2, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99]
Lparam = []
mape = []
for e, p in enumerate(param):
    pipe_mlp = make_pipeline(StandardScaler(),
                                #PCA(n_components=i),
```

```
                                MLPRegressor(hidden_layer_sizes=
hidden_layer_sizes,
                                             activation=activation,
                                             random_state=random_state,
                                             max_iter=max_iter,
                                             alpha = alpha,
                                             learning_rate_init =
learning_rate_init,
                                             shuffle = shuffle,
                                             beta_1 = p,
                                             beta_2 = beta_2,
                                             epsilon = epsilon,

n_iter_no_change=n_iter_no_change,
                                             solver=solver,
                                             verbose = verbose
                                         )
                                )

    pipe_mlp.fit(X_train, y_train)
    y_pred = pipe_mlp.predict(X_test)
    m = mean_absolute_percentage_error(y_test,y_pred)
    mape.append(m)
    Lparam.append(str(p))
    print('mape','|',m)
graf_bar(Lparam,mape,'beta_1', 0)

"""##Beta 2"""

param = [0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 0.995, 0.999, 0.9999]
Lparam = []
mape = []
for e, p in enumerate(param):
    pipe_mlp = make_pipeline(StandardScaler(),
                                #PCA(n_components=i),
                                MLPRegressor(hidden_layer_sizes=
hidden_layer_sizes,
                                             activation=activation,
                                             random_state=random_state,
                                             max_iter=max_iter,
                                             alpha = alpha,
                                             learning_rate_init =
learning_rate_init,
                                             shuffle = shuffle,
                                             beta_1 = beta_1,
                                             beta_2 = p,
                                             epsilon = epsilon,

n_iter_no_change=n_iter_no_change,
                                             solver=solver,
                                             verbose = verbose
                                         )
                                )

    pipe_mlp.fit(X_train, y_train)
    y_pred = pipe_mlp.predict(X_test)
    m = mean_absolute_percentage_error(y_test,y_pred)
    mape.append(m)
    Lparam.append(str(p))
    print('mape','|',m)
graf_bar(Lparam,mape,'beta_2', 0)

param = [1e-10, 1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3]
```

```
Lparam = []
mape = []
for e, p in enumerate(param):
    pipe_mlp = make_pipeline(StandardScaler(),
                             #PCA(n_components=i),
                             MLPRegressor(hidden_layer_sizes=
hidden_layer_sizes,
                                          activation=activation,
                                          random_state=random_state,
                                          max_iter=max_iter,
                                          alpha = alpha,
                                          learning_rate_init =
learning_rate_init,
                                          shuffle = shuffle,
                                          beta_1 = beta_1,
                                          beta_2 = beta_2,
                                          epsilon = p,

n_iter_no_change=n_iter_no_change,
                                          solver=solver,
                                          verbose = verbose
                                      )
                             )

    pipe_mlp.fit(X_train, y_train)
    y_pred = pipe_mlp.predict(X_test)
    m = mean_absolute_percentage_error(y_test,y_pred)
    mape.append(m)
    Lparam.append(str(p))
    print('mape','|',m)
graf_bar(Lparam,mape,'epsilon', 0)

param = [1,10,20,30,40,50,70,100,200,1000,10000,100000]
Lparam = []
mape = []
for e, p in enumerate(param):
    pipe_mlp = make_pipeline(StandardScaler(),
                             #PCA(n_components=i),
                             MLPRegressor(hidden_layer_sizes=
hidden_layer_sizes,
                                          activation=activation,
                                          random_state=random_state,
                                          max_iter=max_iter,
                                          alpha = alpha,
                                          learning_rate_init =
learning_rate_init,
                                          shuffle = shuffle,
                                          beta_1 = beta_1,
                                          beta_2 = beta_2,
                                          epsilon = epsilon,
                                          n_iter_no_change=p,
                                          solver=solver,
                                          verbose = verbose
                                      )
                             )

    pipe_mlp.fit(X_train, y_train)
    y_pred = pipe_mlp.predict(X_test)
    m = mean_absolute_percentage_error(y_test,y_pred)
    mape.append(m)
    Lparam.append(str(p))
    print('mape','|',m)
graf_bar(Lparam,mape,'n_iter_no_change', 0)
```

```
"""#Comparativo

##MLP
"""

pipe_mlp1 = make_pipeline(StandardScaler(),
                          #PCA(n_components=i),
                          MLPRegressor(hidden_layer_sizes= (8,16,32,4),
                                       activation=activation,
                                       random_state=random_state,
                                       max_iter=max_iter,
                                       alpha = alpha,
                                       learning_rate_init =
learning_rate_init,
                                       shuffle = shuffle,
                                       beta_1 = beta_1,
                                       beta_2 = beta_2,
                                       epsilon = epsilon,
                                       n_iter_no_change=p,
                                       solver=solver,
                                       verbose = verbose
                                      )
                          )

pipe_mlp1.fit(X_train, y_train)
y_pred1 = pipe_mlp1.predict(X_test)
m = mean_absolute_percentage_error(y_test,y_pred1)
print('mape','|',m)

"""##MLP+GA"""

pipe_mlp2 = make_pipeline(StandardScaler(),
                          #PCA(n_components=i),
                          MLPRegressor(hidden_layer_sizes= (2,3,59,38),
                                       activation=activation,
                                       random_state=random_state,
                                       max_iter=max_iter,
                                       alpha = alpha,
                                       learning_rate_init =
learning_rate_init,
                                       shuffle = shuffle,
                                       beta_1 = beta_1,
                                       beta_2 = beta_2,
                                       epsilon = epsilon,
                                       n_iter_no_change=p,
                                       solver=solver,
                                       verbose = verbose
                                      )
                          )

pipe_mlp2.fit(X_train, y_train)
y_pred2 = pipe_mlp2.predict(X_test)
m = mean_absolute_percentage_error(y_test,y_pred2)
print('mape','|',m)

"""##Graficos Comparativos"""

df['ConsumoMLP'] = 0
df['ConsumoMLP2'] = 0
y_pred_serie = (np.zeros(60)*np.nan).tolist()+y_pred.tolist()
y_pred2_serie = (np.zeros(60)*np.nan).tolist()+y_pred2.tolist()
df['ConsumoMLP'] = y_pred_serie
```

```
df['ConsumoMLP2'] = y_pred2_serie
df

dcom = df[['Ano Mês','Consumo Médio' ,'Consumo', 'Rlin',
'Consumo_fuzzy','ConsumoMLP','ConsumoMLP2']]
dcom2018 = dcom.iloc[-12:,:]

dcom.info()

import matplotlib.ticker    as ticker
def formatador_de_milhares(valor, p):
    valor = f"{valor:,.0f}"
    mapa_de_traducao = str.maketrans(',.', '.,')
    return valor.translate(mapa_de_traducao)

sns.set_theme()
#style.use(DefStyle)
plt.figure(figsize = (16,10))

ax1 = sns.lineplot(dcom['Ano Mês'],dcom['Consumo'], label = 'Real', palette
= 'Dark2')
#ax2 = sns.lineplot(dcom['Ano Mês'],dcom['ConsumoMLP'], label =
'MLP',palette = 'Dark2')
ax2 = sns.lineplot(dcom['Ano Mês'],dcom['ConsumoMLP2'], label =
'MLP+AG',palette = 'Dark2')

plt.legend()
plt.grid('-')
#plt.title('Perfil de consumo Rural no RS Previsto vs Medido', fontsize=17,
y=1.05)
ax2.yaxis.set_major_formatter(ticker.FuncFormatter(formatador_de_milhares))
plt.ylabel('Consumo [MWh]')
ax2.set_xticks(ax2.get_xticks()[::3])
plt.xticks(rotation = 90, fontsize=10)

plt.show()

import matplotlib.ticker    as ticker
def formatador_de_milhares(valor, p):
    valor = f"{valor:,.0f}"
    mapa_de_traducao = str.maketrans(',.', '.,')
    return valor.translate(mapa_de_traducao)

sns.set_theme()
#style.use(DefStyle)
plt.figure(figsize = (16,10))

ax1 = sns.lineplot(dcom['Ano Mês'],dcom['Consumo'], label = 'Real', palette
= 'Dark2')
ax2 = sns.lineplot(dcom2018['Ano Mês'],dcom2018['ConsumoMLP2'], label =
'MLP2',palette = 'Dark2')
ax3 = sns.lineplot(dcom['Ano Mês'],dcom['Rlin'], label = 'Rlin',palette =
'Dark2')
ax4 = sns.lineplot(dcom2018['Ano Mês'],dcom2018['Consumo_fuzzy'], label =
'Fuzzy',palette = 'Dark2')
ax5 = sns.lineplot(dcom['Ano Mês'],dcom['Consumo Médio'], label =
'Médio',palette = 'Dark2')
ax6 = sns.lineplot(dcom2018['Ano Mês'],dcom2018['ConsumoMLP'], label =
'MLP',palette = 'Dark2')
ax6.lines[2].set_linestyle(':')
ax6.lines[3].set_linestyle(':')
ax6.lines[4].set_linestyle(':')
ax6.lines[5].set_linestyle(':')
```

```
plt.legend()
plt.grid('-')
#plt.title('Perfil de consumo Rural no RS Previsto vs Medido', fontsize=17,
y=1.05)
ax5.yaxis.set_major_formatter(ticker.FuncFormatter(formatador_de_milhares))
plt.ylabel('Consumo [MWh]')
ax6.set_xticks(ax6.get_xticks()[::3])
plt.xticks(rotation = 90, fontsize=10)

plt.show();

dcom2018.columns

print('Comparativo de erros por método horizonte 2018')

print('MLP + GA =' ,
mean_absolute_percentage_error(dcom2018['Consumo'],dcom2018['ConsumoMLP2'])
)
print('MLP      =' ,
mean_absolute_percentage_error(dcom2018['Consumo'],dcom2018['ConsumoMLP']))
print('Fuzzy    =' ,
mean_absolute_percentage_error(dcom2018['Consumo'],dcom2018['Consumo_fuzzy'
]))
print('Rlin     =' ,
mean_absolute_percentage_error(dcom2018['Consumo'],dcom2018['Rlin']))
print('Médio    =' ,
mean_absolute_percentage_error(dcom2018['Consumo'],dcom2018['Consumo
Médio']))

dcom2018.to_csv('/tmp/dfConsumoPrev2018.csv', index= False)
df.to_csv('/tmp/dfConsumoClima3.csv', index= False)

"""#fim"""
```

Algoritmo de Otimização Genética e Rede LSTM:

```
"""LSTM_GEN_e32_p16_BR_res_ibcbr_ibov_obitos_ipca_2015_2020.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/github/lvb86/PD_LSTM_GA/blob/main/code/LSTM_GEN_e32_p16_BR_res_ibcbr_ibov_obitos_ipca_2015_2020.ipynb

#Algoritimo Genético Para Sintonia de LSTM

Aplicado a Scenário:
* BR padrão 32 épocas, pop = 16
* Período 2015-2020

##_Entradas_

* IBCBR
* IBOV
* IPCA
* Obitos
* Clima
* Consumo Resenha EPE:
    - Comercial
    - Industrial
    - Residencial
    - Outros

##Declarações Globais
"""

!pip install -U scikit-learn

if 0:
    from google.colab import drive
    drive.mount('/content/drive')

import matplotlib.pyplot        as plt
import matplotlib.ticker        as ticker
import numpy                    as np
import pandas                   as pd
import seaborn                  as sns
import tensorflow               as tf
import os
import glob
import shutil
import random                   as rn
import math

from datetime                   import datetime, timedelta
from sklearn.model_selection    import
train_test_split,RepeatedStratifiedKFold
from sklearn.preprocessing      import MinMaxScaler
from sklearn.preprocessing      import StandardScaler       #Normalização
dos dados
from sklearn.pipeline           import make_pipeline        #Pipe Line
from sklearn.neural_network     import MLPRegressor
from sklearn.metrics            import mean_absolute_percentage_error

from sklearn.model_selection    import GridSearchCV
#from sklearn.svm import SVC
```

```
from tensorflow                 import keras
from tensorflow.keras           import Sequential, layers, callbacks
from tensorflow.keras           import backend as K
from tensorflow.keras.layers    import Dense, LSTM, Dropout, Bidirectional

seed = 170696
sns.set()
sns.set_theme()

"""#### Path"""

path  = '/tmp/'
patha = path  + 'arcaBR_sn/'
pathb = patha + 'bkp/'
urla  = patha + 'arca.csv'

prefGo    = 'https://docs.google.com/uc?export=download&id='

urld        = prefGo + '1oN489-qxjCNJwUlLrO6sAUSozGzTaG7C'
urlibc      = prefGo + '1QrbgeR7TyHbcNx3l-ke2RD33kn0p0BgY'
urlTabnet   = prefGo + '1z1vn0D9Efnl-wRO_tYDZafZhLGRrPwc0'
urlEPE      = prefGo + '1Qb8aIVyaUpvSn_l6-tbDRtIkUv5RANvN'
urlclima    = prefGo + '1m11IcEh6gNRZC_uuAozBvQp1Ji4c0YvQ'
urlibov     = prefGo + '1S2SQ98qk3V52LnYde04QR2k-X8VhZ_n6'
urlipca     = prefGo + '1l6wRFAprymsQoVoNqAvv1g2EMN2tmVMa'

"""### Randon Freeze"""

def imports():
    import matplotlib.pyplot        as plt
    import matplotlib.ticker        as ticker
    import numpy                    as np
    import pandas                   as pd
    import seaborn                  as sns
    import tensorflow               as tf
    import os
    import glob
    import shutil
    import random                   as rn
    import math

    from datetime                   import datetime, timedelta
    from sklearn.model_selection    import
train_test_split,RepeatedStratifiedKFold
    from sklearn.preprocessing      import MinMaxScaler
    from sklearn.preprocessing      import StandardScaler
#Normalização dos dados
    from sklearn.pipeline           import make_pipeline         #Pipe Line
    from sklearn.neural_network     import MLPRegressor
    from sklearn.metrics            import mean_absolute_percentage_error

    from sklearn.model_selection    import GridSearchCV
    #from sklearn.svm import SVC
    from tensorflow                 import keras
    from tensorflow.keras           import Sequential, layers, callbacks
    from tensorflow.keras           import backend as K
    from tensorflow.keras.layers    import Dense, LSTM, Dropout,
Bidirectional

    seed = 170696
    sns.set()
    sns.set_theme()
```

```
# Utilizando o essa foi a unica maneira encontrada para congelar o randon e
# garantir a reprodutibilidade do modelo

def reset_seed(seed,keras=False, verbose = False):
    imports()
    os.environ['PYTHONHASHSEED'] = '0'
    np.random.seed(seed)
    rn.seed(seed)

    session_conf = tf.compat.v1.ConfigProto(intra_op_parallelism_threads=1,
inter_op_parallelism_threads=1)
    from keras import backend as K
    tf.compat.v1.set_random_seed(seed)
    sess = tf.compat.v1.Session(graph=tf.compat.v1.get_default_graph(),
config=session_conf)
    K.set_session(sess)

def set_seed_rand(verbose = False):

    from datetime import datetime

    dt_seg_now = datetime.now().strftime("%S")
    #print("\t\tSEED =", dt_seg_now)
    nseed = int(dt_seg_now)
    os.environ['PYTHONHASHSEED'] = str(nseed)
    np.random.seed(nseed)
    rn.seed(nseed)
    if verbose: print("\t\tSEED =", dt_seg_now)

"""## Dados"""

anoIni = 2015
anoFim = 2020

ibov = pd.read_csv(urlibov, sep=';', decimal=',', thousands='.',
skiprows=1, skip_blank_lines=True, header=0, encoding='Latin-1')

ibov['mm'] = '0' + ibov['Mês'].astype(str)
ibov['Ano Mês'] = ibov.Ano.astype(str) + '-' +ibov.mm.str[-2:]
ibov = ibov[(ibov.Ano.astype(int) >= anoIni )*(ibov.Ano.astype(int) <=
anoFim)]
ibov.rename(columns={'Valor':'ibov'}, inplace = True)
ibov.head(2)

ibov.info()

ibcbr = pd.read_csv(urlibc, sep =';' ,decimal=',',parse_dates=True,
dayfirst=True,encoding='UTF8', index_col='data' )
ibcbr = ibcbr[(ibcbr.index.year >= anoIni )*(ibcbr.index.year <= anoFim)]
ibcbr.shape

ibcbr['Ano Mês']= ibcbr.index.year.astype(str) +'-'+ ('0' +
ibcbr.index.month.astype(str)).str[-2:]
ibcbr = ibcbr.rename(columns={'valor':'ibcbr'})

ipca = pd.read_csv(urlipca, sep =';' ,decimal=',',parse_dates=True,
dayfirst=True,encoding='UTF8', index_col='data' )
ipca = ipca[(ipca.index.year >= anoIni )*(ipca.index.year <= anoFim)]
ipca['Ano Mês'] = ipca.index.year.astype(str)+'-'+ ('0' +
ipca.index.month.astype(str)).str[-2:]
ipca.rename(columns={'valor':'ipca'}, inplace = True)
ipca.shape
```

```
tabnet = pd.read_csv(urlTabnet, sep =';', decimal=',', skiprows=3,
skipfooter=6, encoding='latin_1')
tabnet['Ano/mês processamento'] = tabnet['Ano/mês
processamento'].str.replace('..','',regex=False)
tabnet = tabnet[tabnet['Ano/mês processamento'].str.len() >= 6]
tabnet['Ano'] = tabnet['Ano/mês processamento'].str[-4:]
tabnet['Mês'] = tabnet['Ano/mês processamento'].str[:-5]
tabnet['Mês'] = tabnet['Mês'].str.strip()
mapmes = {'Janeiro':'01', 'Fevereiro':'02', 'Março':'03', 'Abril':'04',
'Maio':'05', 'Junho':'06',
 'Julho':'07', 'Agosto':'08', 'Setembro':'09', 'Outubro':'10',
'Novembro':'11', 'Dezembro':'12'}
tabnet['mm'] = tabnet['Mês'].map(mapmes)
tabnet.drop(columns = 'Ano/mês processamento')
tabnet['Ano Mês'] = tabnet.Ano +'-'+ tabnet.mm
tabnet = tabnet.rename(columns = {'Total': 'BR'})
obtosBR = tabnet[tabnet.Ano.astype(int) <= anoFim][['Ano Mês', 'BR']]
obtosBR = obtosBR.rename(columns = {'BR':'Obitos'})
obtosBR.shape

pd.read_csv(urlEPE)

dfEPE = pd.read_csv(urlEPE)
dfEPE = dfEPE[(dfEPE.Ano >= anoIni) & (dfEPE.Ano <= anoFim)]
dfEPE = dfEPE.pivot(index = ['Ano Mês','Mes'], columns='Setor',values =
'Total')
dfEPE['Consumo'] = dfEPE.Comercial + dfEPE.Residencial + dfEPE.Comercial +
dfEPE.Outros

clima = pd.read_csv(urlclima, index_col=False)

df = dfEPE.reset_index()
df = pd.merge(df,ibcbr[['Ano Mês','ibcbr']],left_on='Ano Mês',right_on='Ano
Mês')
df = pd.merge(df,ibov[['Ano Mês','ibov']],left_on='Ano Mês',right_on='Ano
Mês')
df = pd.merge(df,obtosBR[['Ano Mês','Obitos']],left_on='Ano
Mês',right_on='Ano Mês')
df = pd.merge(df,ipca[['Ano Mês','ipca']],left_on='Ano Mês',right_on='Ano
Mês')
df = pd.merge(df,clima.reset_index()[['Ano Mês','Dias com
Precipitação','Precipitação', 'Pressão', 'Temperatura', 'Vento Máx', 'Vento
Méd' ]],              left_on='Ano Mês',right_on='Ano Mês')

df.head(3)

df.columns

df['mes_sin'] = np.sin(df['Mes']*2*np.pi/12)
df['mes_cos'] = np.cos(df['Mes']*2*np.pi/12)
d = df.drop(columns=['Ano Mês', 'Mes']).reset_index()
Ycolumns = ['Consumo', 'Comercial', 'Industrial', 'Residencial', 'Outros']
Xcolumns = [*{*d.columns.to_list()}-{*Ycolumns,'index','level_0'}]

print('X\n',len(Xcolumns),'\n', Xcolumns)
print('Y\n',len(Ycolumns),'\n', Ycolumns)

"""## Funções

#### Train Test Split
"""

df.shape[0]
```

```
def TrainTestData(col_i):
    lin = df.shape[0]
    lin_train = lin - 12
    X = d[Xcolumns]
    y = d[Ycolumns]
    y = y.iloc[:,col_i]
    # Different scaler for input and output
    scaler_x = MinMaxScaler(feature_range = (0,1))
    scaler_y = MinMaxScaler(feature_range = (0,1))
    #print(2)
    # Fit the scaler using available training data
    input_scaler = scaler_x.fit(X)
    X_norm = input_scaler.transform(X)
    #print('y_shape= ',y.shape)
    #print('y_shape= ',len(y.shape))
    if len(y.shape) == 1:
       # print('..reshape')
        output_scaler = scaler_y.fit(y.to_numpy().reshape(-1,1))
        y_norm = output_scaler.transform(y.to_numpy().reshape(-1,1))
    else:
        output_scaler = scaler_y.fit(y)
        y_norm = output_scaler.transform(y)

    X_train = X_norm[:lin_train]
    y_train = y_norm[:lin_train]

    X_test = X_norm[lin_train:]
    y_test = y_norm[lin_train:]

    y_real = y[lin_train:]
    X_train = X_train.reshape(X_train.shape[0], 1, X_train.shape[1])
    X_test = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])

    #print("Dimensões X: ",'X_train',X_train.shape,'X_test', X_test.shape)
    #print("Dimensões Y: ",'Y_train',y_train.shape,'Y_test', y_test.shape)

    shape_in = X_train.shape[2]
    shape_out = y_train.shape[1]

    return shape_in, shape_out, X_train, y_train, X_test, y_test, y_real,
input_scaler, output_scaler

def TrainTestData_old(verbose=False):
    lin = df.shape[0]
    lin_train = lin - 12
    X = d[Xcolumns]
    y = d[Ycolumns]
    # Different scaler for input and output
    scaler_x = MinMaxScaler(feature_range = (0,1))
    scaler_y = MinMaxScaler(feature_range = (0,1))

    # Fit the scaler using available training data
    input_scaler = scaler_x.fit(X)
    #output_scaler = scaler_y.fit(y.to_numpy().reshape(-1,1))
    output_scaler = scaler_y.fit(y)

    # Apply the scaler to training data
    X_norm = input_scaler.transform(X)
    #y_norm = output_scaler.transform(y.to_numpy().reshape(-1,1))
    y_norm = output_scaler.transform(y)

    X_train = X_norm[:lin_train]
```

```
    y_train = y_norm[:lin_train]

    X_test = X_norm[lin_train:]
    y_test = y_norm[lin_train:]

    y_real = y[lin_train:]

    X_train = X_train.reshape(X_train.shape[0], 1, X_train.shape[1])
    X_test = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])
    if verbose:
        print("Dimensões X: ",'X_train',X_train.shape,'X_test',
X_test.shape)
        print("Dimensões Y: ",'Y_train',y_train.shape,'Y_test',
y_test.shape)

    shape_in = X_train.shape[2]
    shape_out = y_train.shape[1]

    return
X,y,input_scaler,output_scaler,X_train,y_train,X_test,y_test,y_real,shape_i
n, shape_out

"""###Genéticas

#### Inicia Populacão
"""

def inicializa_populacao(pop_tamanho, n_genes, limites):
    """
    Inicializa a população de acordo com o tamanho da população
    e número de genes.

    param pop_tamanho:     Número de individuos na população
    param n_genes:         Número de genes (Variáveis) no problema
    param limites:         Tupla contendo o número mínimo e máximo
permitido
    return:                Um array numpy com a população iniciada
                           randomicamente
    """

    pop0 = np.random.randint(
      limites[0], limites[1], size=(pop_tamanho, n_genes)
    )
    pop=[]
    for e, i in enumerate(pop0):
        if np.sum(i) == 0:
            n = np.random.randint(limites[0]+1, limites[1], size=(1,
n_genes))
            #print(e,i,n)
            pop.append(n)
        else:
            pop.append(i)
    return pop

"""####Função de aptidão"""

def rede_desc(bl_list, unit_list, drop_list):
    '''
    Retorna descrição gráfica da Rede

    '''
    dic_bl = {0: 'LSTM', 1:'BILSTM'}
    desc = '->'
```

```
    for e,i in enumerate(unit_list):
        if i >0:
            if e < 2:
                desc+=(f'{dic_bl[bl_list[e]]}')
            else:
                desc+=('MLP')
            desc+=(f'({unit_list[e]})->')
            if e < 5:
                desc+=(f'd({drop_list[e]})->')
    return desc

def decode_gen(chave, verbose = False):
    '''
    Recebe chave genética e retorna estrutura da rede
    considerando arquitetura predefinida

            Tipo---\ Dropout -----------------\ Neurônios------------------
-\
            B|L  B|L  D12  D23  D34  D45  D56  NL1  NL2  NL3  NL4  NL5  NL6
    chave =[ p01, p02, p03, p04, p05, p06, p07, p07, p08, p09, p10, p11,
p12]
            2|1  2|1  2|1  2|1  2|1

    Tipo      -> se par BILSTM se não LSTM
    Dropout   -> If 0 = 0
                 Else
                     Se par 0.2 senão 0.1
    Neurônios -> If 0 desetiva layer e dropout seguinte
                 Else numero de neurônios da camada
    '''

    bl_list = []
    drop_list = []
    unit_list = []
    Lini = 0
    Lcount = 0

    for i in range(7,13):
        if chave[i] >0:
            ini = i-6
            break
    for i in range(7,13):
        if chave[i] >0:
            Lcount +=1
    if verbose: print(f'\tN Layers = {Lcount}')
    else: print('.',sep='',end='')

    for i in range(2):
        if chave[i] % 2 == 0:
            bl_list.append(1)
        else:
            bl_list.append(0)

    for i in range(2,7):
        if chave[i] ==0:
            drop_list.append(0)
        else:
            if chave[i] % 2 == 0:
                drop_list.append(0.2)
            else:

                drop_list.append(0.1)
```

```
    for i in range(7,13):
        unit_list.append(chave[i])

    desc = rede_desc(bl_list, unit_list, drop_list)
    if verbose: print('\t'+desc)
    else: print('.',sep='',end='')
    return bl_list, unit_list, drop_list, desc

def monta_modelo(bl_list, unit_list, drop_list):
    '''
    Modelo Keras
    '''
    model = tf.keras.models.Sequential()
    model.add(tf.keras.layers.InputLayer(input_shape=(1,shape_in)))

    for e,i in enumerate(unit_list):
        if i > 0:
            if e < 2:           # LSTM
                if bl_list[e]:
                    model.add(tf.keras.layers.Bidirectional(LSTM (units =
i, return_sequences=True )))
                else:
                    model.add(tf.keras.layers.LSTM(units = i,
return_sequences=True ))

            else:               #MLP
                model.add(tf.keras.layers.Dense(units = i,
activation=tf.nn.relu) )
            if e < 5:
                                #Dropout
                model.add(tf.keras.layers.Dropout(drop_list[e], seed =
seed) )
    model.add(tf.keras.layers.Dense(shape_out))

    return model

def func_aptidao(individual, verbose = False):
    """
    Calcula a aptidão de cada individuo

    :param individual:   Cromosomo de genes representando um individuo
    :return:             A aptidão individual e MAPE
    """
    reset_seed(seed)    #Para Garantir a repetibilidae do modelo

    bl_list, unit_list, drop_list, desc = decode_gen(individual)
    model = monta_modelo(bl_list, unit_list, drop_list)

    with tf.device(':CPU:0'):

        model.compile(optimizer = 'adam',loss='mse',
metrics=['accuracy','mape'])
        early_stop = keras.callbacks.EarlyStopping(monitor = 'loss',
                                                   patience = 10)
        model.fit(  x=X_train,
                    y=y_train,
                    batch_size=None,
                    epochs=10000,
                    verbose=False,
                    callbacks=[early_stop],
                    validation_split=0.0,
                    #validation_data=(X_test,y_test),
                    shuffle=False,
```

```
                    class_weight=None,
                    sample_weight=None,
                    initial_epoch=0,
                    steps_per_epoch=19,
                    validation_steps=1,
                    validation_batch_size=None,
                    validation_freq=1,
                    max_queue_size=10,
                    workers=1,
                    use_multiprocessing=False)
        y_pred_norm = model.predict(X_test)

        if shape_out ==1:
            y_pred =
output_scaler.inverse_transform(y_pred_norm.ravel().reshape(-1, 1))
        else:
            y_pred =
output_scaler.inverse_transform(y_pred_norm.reshape(12,shape_out))

        mape = mean_absolute_percentage_error(y_real,y_pred)
        apti = 1-mape

    if verbose: print('\t\t','mape','|',mape, '||', 'aptidão','|',apti)
    else: print('.',sep='',end='')
    set_seed_rand()
    yp = y_pred.reshape(12,shape_out)
    return mape, apti, yp

"""#### Seleção de Progenitores"""

def selecao_de_progenitores(individuos, probabilidades, pares, verbose =
False):
    """
    Seleciona os pais de acordo com a estratégia "roulette_wheel" que
seleciona
    individuos aleatoriamente utilizando a maior aptidão como maior
    probabilidade

    param individuos:       Numero de individuos
    param probabilidades:  distribuição de probabilidade
    return:                pares escolhidos aleatoriamente
    """
    from math import ceil
    n = len(individuos)
    pares = int(pares)
    if pares*2 > n:
        pares = n/2

    p1, p2 = None, None
    pn = np.random.choice(range(n),2*pares, replace =
False,p=probabilidades)
    pr = np.random.choice(pn,len(pn),replace=False)
    pp = np.array(pr).reshape(ceil(len(pn)/2),2)

    if verbose: print('\t\t',pp)
    else: print('.',sep='',end='')
    return pp

"""#### Cruzamento"""

def cruzamento(casal, populacao, ng, verbose = False ):
    '''
    param casal:     index tupla (p1,p2) do casal que ira cruzar
```

```
    param populacao: recebe a popopulação atual

    return:          2 filhos com os cruzamentos aleatórios dos genes
    '''
    p1 = casal[0]   #progenitor 1
    p2 = casal[1]   #progenitor 2
    f1 = []         #filho 1
    f2 = []         #filho 2
    ng = populacao[p1].shape[0]

    if verbose:
        print('\t\tpar --', casal,'------')
        print('\t\tp:',p1,populacao[p1]) #p1
        print('\t\tp:',p2,populacao[p2]) #p2
    else: print('.',sep='',end='')

    s1=np.ones(ng)
    while(s1.sum()==ng or s1.sum()==0):   # garantia de que havera ao menos
        s1=np.random.choice([0,1],ng)    # 1 cruzamento
    if verbose: print('\t\txxx ', s1)
    else: print('.',sep='',end='')
    for i, b in enumerate(s1):
        #print(i,b)
        if b:
            #print(x[7][i])
            f1.append(populacao[p1][i])
            f2.append(populacao[p2][i])
        else:
            #print(x[9][i])
            f1.append(populacao[p2][i])
            f2.append(populacao[p1][i])
    if verbose:
        print('\t\tf1  ',f1)
        print('\t\tf2  ',f2)
    else: print('.',sep='',end='')
    return np.array(f1),np.array(f2)

"""#### Mutação"""

#mutacao
def mutacao(original, proba_mut, ng, verbose= False):
    '''
    apartir de um individuo original será gerado 1 individuo mutante

    param original:     cromossomo do individuo original
    param proba_mut:    probabilidade de mutação
    return:             mutante
    '''
    mutante = []
    if verbose: print('\t\tori ', original)
    else: print('.',sep='',end='')

    gm = np.random.choice(64,ng)    #Sequencia mutante Aleatória

    s1=np.ones(ng)
    while(s1.sum()==0):   # garantia de que havera ao menos
        s1=np.random.choice([0,1],ng,p=[1-proba_mut,proba_mut])    # 1
mutação
    if verbose: print('\t\txxx ', s1)
    else: print('.',sep='',end='')
    for i, b in enumerate(s1):
        #print(i,b)
        if b:
```

```
            #print(x[7][i])
            mutante.append(gm[i])

        else:
            mutante.append(original[i])
    if verbose: print('\t\tmut ',mutante)
    else: print('.',sep='',end='')
    return  np.array(mutante)

"""###Busca e Salvamento """

def busca_item_lista(_item, _lista):
    '''
    Percorre a lista de arrays e retorna a posição se encontrar ou false se
    não enontrar

    '''
    for index, it in enumerate(_lista):

        #print(index,it,_item, all(it==_item))
        if (all(it==_item)):
            return True, index
    return False, False

def salva_arca(_ep,idtest):
    now = datetime.now().strftime('_%y_%m_%d-%H_%M_')
    url = patha + 'arca' +str(idtest)+ now + str(_ep) + '.csv'
    #arca = pd.DataFrame([arca_gen,arca_mape,arca_apt]).T
    arca = pd.DataFrame(dic_arca).T.iloc[:,-3:]
    arca.columns = ['gen','mape','apt']
    arca.apt = 1-arca.mape  #correção de aptidão relativa para geral
    arca.to_csv(url,index=False)

def recupera_arca():
    '''
    Recupera dados de testes passados

    return: arca_gen, arca_mape, arca_apt, dic_arca, dic_arca_i
    '''
    url = patha + 'arca.csv'
    arca_gen_str = pd.read_csv(url).gen.to_list()
    arca_mape    = pd.read_csv(url).mape.to_list()
    arca_apt     = pd.read_csv(url).apt.to_list()

    arca_gen = []
    for e, arc_g in enumerate(arca_gen_str):
        arca_gen.append(gene_str2numpy(arc_g))
    print(len(arca_gen))
    dic_arca = {} # Cria dicionário vazio. # set() = conjunto vazio
    dic_arca_i = {}
    for e, i in enumerate(arca_gen):
        dic_arca[tuple(i)] = [e,i,arca_mape[e],arca_apt[e]]
        dic_arca_i[e] = i

    return arca_gen, arca_mape, arca_apt, dic_arca, dic_arca_i

#arca_gen, arca_mape, arca_apt, dic_arca, dic_arca_i = recupera_arca()

#len(arca_gen) == len(arca_mape)

def gene_str2numpy(_str):
    '''
```

```
    return: np.array com a sequencialida
    '''
    indice = []
    arn = []
    ultimo = 9999
    for i, c in enumerate(_str):
        if ord(c) in (91,32,93):
            if i != ultimo + 1:
                indice.append(i)
            # print(i,c,ord(c),)
                ultimo = i

    for i in range(len(indice)-1):
        n_str =_str[indice[i]+1:indice[i+1]]
        arn.append(int(n_str.strip()))
    return np.array(arn)

def salva_resultado(_ep,_res,_mut,_tempo,_pop,_pop_r,_pop_c,idtest):
    now = datetime.now().strftime('_%y_%m_%d-%H_%M_')
    url = path + 'resultado' +str(idtest)+ now + str(_ep) + '.csv'
    result = pd.DataFrame([_res,_mut,_tempo,_pop,_pop_r,_pop_c]).T
    result.columns =
['Mape','Mutacao','Tempo','Populacao','Pop_Recuperada','Pop_Calculada']
    result.to_csv(url,index = False)

def atualiza_arca(_bol):
    if _bol:
        os.chdir(patha)
        extension = 'csv'
        all_filenames = [i for i in
glob.glob('arca*.{}'.format(extension))]
        if len(all_filenames):
            now = datetime.now().strftime('%y_%m_%d-%H_%M')
            url = pathb + 'arca_bkp_' + now + '.csv'
            if os.path.exists(patha+'arca.csv'):
                pd.read_csv(patha+'arca.csv').to_csv(url)
            #combine all files in the list
            combined = pd.concat([pd.read_csv(f) for f in all_filenames ])
            filtred =
combined[['gen','mape','apt']].sort_values('mape').drop_duplicates()
            #export to csv
            filtred.to_csv("arca.csv", index=False, encoding='utf-8-sig')
            # move arquivos para o bkp
            for f in all_filenames:
                if f !='arca.csv':
                    shutil.move(patha+f,pathb+f)
                    print(f)

#https://www.freecodecamp.org/news/how-to-combine-multiple-csv-files-with-
8-lines-of-code-265183e0854/
#https://datatofish.com/move-file-python/

if 0:
    dfa = pd.read_csv(patha + 'arca.csv')
    dfa['len'] = dfa['gen'].apply(lambda x: len(x))
    #dfa['len'].count_values()
    dfb=dfa[dfa['len'] == 40]
    dfb.drop(['len'], axis =1 ,inplace = True)
    dfb.sort_values('mape').to_csv(patha + 'arca.csv', index = False)
    pd.read_csv(patha + 'arca.csv')

def df_arca_hist(df):
    '''
```

```
    Gera dados únicos para comparativo em histograma, considerando que:
    (3, 2, 0, 1) = (3, 2, 1)
    uma vez que na lógica implementada para arquiterura MLP a camada com 0
    neurônios é suprimida.
    '''
    arca_hist = df[['gen','mape']]
    arca_hist['gen'] = arca_hist.gen.apply(lambda g: gene_str2numpy(g))
    arca_hist['dim'] = arca_hist.gen.apply(lambda x: np.shape(x)[0])
    arca_hist['gen'] = arca_hist.gen.apply(lambda x: str(x[np.where(x)]))
    arca_hist.drop_duplicates(inplace=True)

    return arca_hist[arca_hist.dim != 0]

"""###Relatórios e Medidas"""

def time_diff(T1, verbose = False):
    T2 = datetime.now()
    format = '%H:%M'
    #tdiff = datetime.strptime(T1, format) - datetime.strptime(T2, format)
    tdiff = T2 - T1
    #if tdiff. < 0:
    #    tdiff = timedelta(days = 0,
    #                seconds = tdiff.seconds, microseconds =
tdiff.microseconds)
    tdiff.total_seconds
    if verbose: print( '....tempo transcorrido ', tdiff)
    return tdiff.total_seconds()/60
#

def formatador_de_milhares(valor, p):
        valor = f"{valor:,.0f}"
        mapa_de_traducao = str.maketrans(',.', '.,')
        return valor.translate(mapa_de_traducao)

"""#### Relatório Parcial"""

def relatorio_parcial(ep,
                      count_parada,
                      arca_gen,
                      ar_mape_gen,
                      ar_mape_min,
                      ar_mape_ind,

                      ge_mape_gen,
                      ge_mape_min,
                      ge_mape_ind,

                      list_tdiff,
                      list_t_pop,
                      list_t_pop_c,
                      list_t_pop_r,
                      list_mutacao,
                      max_comb,
                      res,
                      teste_mape
                      ):
    print('____________________RELATÓRIO
PARCIAL_____________________________'
    ,'\n\t\t Geração                        =', ep
    ,'\n\t\t Parado                         =', count_parada
    ,'\n\t\t Códigos na arca                =', len(arca_gen)
    ,'\n\t\t % da população avaliada       =',
f'{(len(arca_gen)/max_comb)*100:.5f} %'
```

```
    ,'\n\t\t Tempo médio por geração    =', f'{np.mean(list_tdiff):.5f} %'
    ,'\n\t\t Tempo médio por ind_c      =',
f'{sum(list_tdiff)/sum(list_t_pop):.2f} min'
    ,'\n-->Nesta
Geração:_____________________________________________'
    ,'\n\t\t Novos indivíduos Testados  =', sum(list_t_pop_c)
    ,'|'                                 ,
f'{(sum(list_t_pop_c)/max_comb)*100:.5f} %'
    ,'\n\t\t Clones indivíduos Testados =', sum(list_t_pop_r)
    ,'\n\t\t Total indivíduos Testados  =', sum(list_t_pop)
    ,'|'                                 ,
f'{(sum(list_t_pop)/max_comb)*100.:.5f} %'
    #,'\n\t\t Tempo por indivíduos novo  =',
f'{sum(list_tdiff)/sum(list_t_pop_c):.2f} min'
    #,'\n\t\t %(indivíduos novo/Testados)=',
f'{(sum(list_t_pop_c)/max_comb)*100:.5} %'
    )

    _,_,_, desc1 = decode_gen(ge_mape_gen)
    _,_,_, desc2 = decode_gen(ar_mape_gen)

    print('--------Melhor desta Geração ----------',
          '\n\t\t MAPE      = ', round(ge_mape_min,6),
          '\n\t\t id_arca   = ', ge_mape_ind         ,
          '\n\t\t gen       = ', ge_mape_gen         ,
          '\n\t'               , desc1
          )

    print('------------Melhor testado ------------',
          '\n\t\t MAPE      = ', round(ar_mape_min,6),
          '\n\t\t id_arca   = ', ar_mape_ind         ,
          '\n\t\t gen       = ', ar_mape_gen         ,
          '\n\t'               , desc2               ,
          )
    fig, axs = plt.subplots(3,2, figsize=(12,15))

    sns.lineplot(ax=axs[0,0],x=range(len(res)),y=res)
    axs[0,0].set_title('MAPE mínimo por Geração')
    axs[0,0].set_ylabel('MAPE')
    #axs[0,0].set_xlabel('Geração')

    sns.lineplot(ax=axs[1,0],x=range(len(list_mutacao)),y=list_mutacao)
    axs[1,0].set_title('Evolução do Fator de mutação')
    axs[1,0].set_ylabel('Fator de mutação')
    #axs[1,0].set_xlabel('Geração')

    sns.lineplot(ax=axs[0,1],x=range(len(list_t_pop)),y=list_t_pop, label =
'total')
    sns.lineplot(ax=axs[0,1],x=range(len(list_t_pop_c)),y=list_t_pop_c,
label = 'calculada')
    sns.lineplot(ax=axs[0,1],x=range(len(list_t_pop_r)),y=list_t_pop_r,
label = 'recuperada')
    axs[0,1].set_title('Evolução tamanho da população')
    axs[0,1].set_ylabel('Indivíduos')
    #axs[0,1].set_xlabel('Geração')

    sns.lineplot(ax=axs[1,1],x=range(len(list_tdiff)),y=list_tdiff)

sns.lineplot(ax=axs[1,1],x=range(len(list_tdiff)),y=np.mean(list_tdiff))
    axs[1,1].set_title('Perfil de tempo por iteração')
    axs[1,1].set_ylabel('tempo [min]')
    #axs[1,1].set_xlabel('Geração')
```

```
    sns.histplot(ax=axs[2,0],data=teste_mape, kde=True)
    axs[2,0].set_title('Distribuição de erro por genoma teste')

    list_t_pop_a = []
    list_t_pop_ca = []
    list_t_pop_ra = []
    for e,i in enumerate(list_t_pop_c):
        list_t_pop_a.append(sum(list_t_pop[:e+1]))
        list_t_pop_ca.append(sum(list_t_pop_c[:e+1]))
        list_t_pop_ra.append(sum(list_t_pop_r[:e+1]))
    sns.lineplot(ax=axs[2,1],x=range(len(list_t_pop_a)),
                 y=list_t_pop_a,label = 'total')
    sns.lineplot(ax=axs[2,1],x=range(len(list_t_pop_ca)),
                 y=list_t_pop_ca, label = 'calculada')
    sns.lineplot(ax=axs[2,1],x=range(len(list_t_pop_ra)),
                 y=list_t_pop_ra,label = 'recuperada')
    #sns.ecfplot(ax=axs[2,1],list_t_pop_t, label = 'indivíduos avaliados')
    axs[2,1].set_title('População acumulada')
    axs[2,1].set_ylabel('Indivíduos')

    #arca_hist = df_arca_hist(pd.read_csv(urla))
    #sns.histplot(ax=axs[2,1],data=arca_hist, kde=True)
    #axs[2,1].set_title('Distribuição de erro por genoma hist.')

    plt.suptitle('Resultados dos testes Geração = '+str(ep) , y=.93,
fontsize=17)
    plt.show();

"""#### Perfil de Consumo"""

def plt_consumo12(y_true,y_pred):
    plt.figure(figsize=(10,8))
    ax1 = sns.lineplot(x=range(1,13),y=y_true, label='Real')
    ax2 = sns.lineplot(x=range(1,13),y=y_pred.ravel(), label='LSTM')

ax2.yaxis.set_major_formatter(ticker.FuncFormatter(formatador_de_milhares))
    plt.legend()
    plt.grid('-')
    plt.ylabel('Consumo [MWh]')
    #ax2.set_xticks(ax2.get_xticks()[::3])
    #plt.xticks(rotation = 90, fontsize=10)
    plt.show();

def plt_barras(x,y,*titulo):
    plt.figure(figsize=(12,8))
    sns.set_style('darkgrid')
    g = sns.barplot(x=x, y=y, palette='rocket')
    plt.title('MAPE por SetorN1', fontsize = 17)

    for i in range(len(x)):
        g.text(i,y[i]+0.001, round(y[i],3), color='black', ha="center",
fontsize=15)
    plt.xticks(rotation = 90, fontsize=15)
    plt.ylabel('MAPE')
    #plt.xlabel('SetorN1')
    plt.show()
    ;

def plt_perfis_de_consumo(yp):
    list_mape = []

    fig, axs = plt.subplots(3,3, figsize=(12,15))
    for e, i in enumerate(y.columns):
```

```
        if e < 3:
            s = 0
        elif e < 6:
            s = 1
        else:
            s = 2

        sns.lineplot(ax=axs[s,e-s*3],x=range(1,13),y=yp[:,e]/1000, label =
'pred')
        sns.lineplot(ax=axs[s,e-s*3],x=range(1,13),y=y.iloc[60:,e]/1000,
label = 'real')
        axs[s,e-s*3].set_title(i)
        axs[s,e-s*3].set_xlabel('')
        axs[s,e-s*3].set_ylabel('')
        #axs[s,e-s*3].set_ylim(0,3_000_000)
        axs[s,e-s*3].set_ylim(0)
        axs[s,e-s*3].set_xticks(range(0,13,2))
        axs[s,e-s*3].set_xlim(1, 12)
        axs[s,e-
s*3].yaxis.set_major_formatter(ticker.FuncFormatter(formatador_de_milhares)
)
        mape = mean_absolute_percentage_error(y.iloc[60:,e],yp[:,e])
        list_mape.append(mape)
    list_mape.append(np.array(list_mape).mean())

    plt.suptitle('Consumo [kW] por SetorN1' , y=.93, fontsize=17)
    plt.show()
    ;
    if 0:
        y_bars = [*y.columns,'Média']
        plt.figure(figsize=(12,8))
        sns.set_style('darkgrid')
        g = sns.barplot(x=y_bars, y=list_mape, palette='rocket')
        plt.title('MAPE por SetorN1', fontsize = 17)

        for i in range(len(y_bars)):
            g.text(i,list_mape[i]+0.001, round(list_mape[i],3),
color='black', ha="center", fontsize=15)
        plt.xticks(rotation = 90, fontsize=15)
        plt.ylabel('MAPE')
        #plt.xlabel('SetorN1')
        plt.show()
        ;

def plt_perfis_de_consumo2(yp):
    yp = y_pred_list.T
    list_mape = []

    fig, axs = plt.subplots(2,3, figsize=(12,15))
    for e, i in enumerate(y.columns):
        if e < 3:
            s = 0
        elif e < 6:
            s = 1
        else:
            s = 2
        print(e, i)
        sns.lineplot(ax=axs[s,e-s*3],x=range(1,13),y=yp[-12:,e]/1000, label
= 'pred')
        sns.lineplot(ax=axs[s,e-s*3],x=range(1,13),y=y.iloc[-12:,e]/1000,
label = 'real')
        axs[s,e-s*3].set_title(i)
        axs[s,e-s*3].set_xlabel('')
```

```
            axs[s,e-s*3].set_ylabel('')
            #axs[s,e-s*3].set_ylim(0,3_000_000)
            axs[s,e-s*3].set_ylim(0)
            axs[s,e-s*3].set_xticks(range(0,13,2))
            axs[s,e-s*3].set_xlim(1, 12)
            axs[s,e-
s*3].yaxis.set_major_formatter(ticker.FuncFormatter(formatador_de_milhares)
)
            mape = mean_absolute_percentage_error(y.iloc[-12:,e],yp[:,e])
            list_mape.append(mape)
        list_mape.append(np.array(list_mape).mean())

        plt.suptitle('Consumo [kW] por SetorN1' , y=.93, fontsize=17)
        y_bars = [*y.columns,'Média']
        g = sns.barplot(ax=axs[1,2], x=y_bars, y=list_mape, palette='rocket')
        for i in range(len(y_bars)):
            g.text(i,list_mape[i]+0.001, round(list_mape[i],3), color='black',
ha="center", fontsize=10)
        plt.xticks(rotation = 90, fontsize=10)
        plt.title('MAPE por SetorN1', fontsize = 10)

        plt.show()
        ;

"""###Algoritmo Genético"""

def algoritmo_genetico(
                    ini_pop         = True     ,
                    pop_selecao     = 0        ,
                    recup_memo      = False    ,
                    epocas          = 100      ,
                    pop_tamanho     = 64       ,
                    n_genes         = 13       ,
                    limites_genes   = (0,64)   ,
                    indice_mut      = 0.2      ,
                    #casais         = int((pop_tamanho+pop_selecao)/4)   ,
                    max_parado      = 200      ,
                    alvo            = 0.001    ,
                    bool_salva_arca = False    ,
                    verbose         = False
        ):
        '''
        Parâmetros de entrada:
        ini_pop         # Inicializa população aleatória
        pop_selecao     # N melhore hist Selecionados para geração 0
        recup_memo      # True recupera memória de iterações passadas
        epocas          # total de gerações/Epocas
        pop_tamanho     # tamanho da inicial população
        n_genes         # Número de Genes
        limites_genes   # Limite de valores para cada Gene
        indice_mut      # Probaabilidade de Mutação inicial
        #casais         # Número de casais Progenitores
        max_parado      # Limite de epocas parado no mesmo erro
        alvo            # Erro Alvo
        bol_salva_arca  # Exportar resultados
        verbose         # Saída de fluxo

        Saídas
        Código Genético do melhor modelo
        Valor de Erro (MAPE) do melhor modelo
        '''
        #-------------------------------------------------------------------
----
```

```
    ## Variáveis de Controle
    casais          = int((pop_tamanho+pop_selecao)/4)
    arca_gen        = []            # Memória genetica
    arca_apt        = []            # Memória de aptidão
    arca_mape       = []            # Memória de mape
    teste_mape      = []
    count_parada    = 0
    count_pop_r     = 0
    count_pop_c     = 0
    res             = []
    elite           = []
    list_mutacao    = []
    list_tdiff      = []
    list_t_pop      = []
    list_t_pop_r    = []
    list_t_pop_c    = []
    novos_individuos= []
    pre_selecionado = []
    selecionado     = []
    pop             = []
    ep              = 0

    min_anterior  = 1
    test_comb     = epocas*pop_tamanho
    max_comb      = ((limites_genes[1]-limites_genes[0])+1)**n_genes

    if verbose:

        print('->Algoritmo genético de otimização -------------------------
----------')
        print(  '\t Nova Populaçao                          = '  , ini_pop
,
            '\n\t Recupera memória de iterações passadas = '  , recup_memo
,
            '\n\t Gerações máximas                       = '  , epocas
,
            '\n\t Mais aptos selecionados em memória     = '  , pop_selecao
,
            '\n\t Tamanho da população                   = '  , pop_tamanho
,
            '\n\t Número de Genes                        = '  , n_genes
,
            '\n\t Limite de valores para cada Gene       = '  ,
limites_genes ,
            '\n\t Probabilidade de Mutação inicial       = '  , indice_mut
,
            '\n\t Casais por época                       = '  , casais
,
            '\n\t Limite de épocas parado no mesmo erro  = '  , max_parado
,
            '\n\t Erro Alvo                              = '  , alvo
,
            '\n\t Total de combinações no teste          = '  ,
test_comb/20  ,
            '\n\t Tempo de execução estimado em horas    = '  ,
test_comb/20  ,
            '\n\t Total de combinações possíveis         = '  , max_comb
,
            '\n\t Tempo máximo max comb. em anos         = '  ,
max_comb/175200
            )
    else: print('.',sep='',end='')
```

```
    if os.path.isfile(patha+'arca.csv') and recup_memo:
        arca_gen, arca_mape, arca_apt, dic_arca, dic_arca_i=
recupera_arca()

        if verbose:
            print('-------------------------------------------------------
-----------'
            ,'\n\t\t\t\t', len(arca_gen),'codigos recuperados'
            ,'\n\t\t\t\t', f'{(len(arca_gen)/max_comb)*100:.5f} % da
população avaliada' )
        else: print('.',sep='',end='')
    else:
        dic_arca = {}
        dic_arca_i = {}
    if ini_pop:
        if verbose:
            print('-->Inicializa
População.....................................................')
        else: print('.',sep='',end='')
        pop = inicializa_populacao(pop_tamanho,n_genes,limites_genes)
        if verbose: print('População Inicial:',pop)
        else: print('.',sep='',end='')

    if pop_selecao >0:
        for e, i in enumerate(arca_gen):
            if len(i) ==n_genes:
                pre_selecionado.append(i)
        if verbose: print (len(pre_selecionado),'possiveis de seleção')
        else: print('.',sep='',end='')
        for e, i in enumerate(pre_selecionado[0:pop_selecao]):
            if verbose: print (e,i, len(i))
            else: print('.',sep='',end='')
            selecionado.append(i)
        pop += selecionado
    for ep in range(epocas):
        T1 = datetime.now()
        gera_mape = []
        count_pop_r = 0
        count_pop_c = 0
        if verbose:
            print('-->>epoca',ep,'-------------------------------------
-----------')
        else: print('\n-->#ep.',ep,sep='',end='')
        #print('Novos Individuos:',novos_individuos)
        if ep > 0:
            if verbose:
                print('-->>>Seleção----------------------------------------
-----------')
            else: print('.',sep='',end='')
            for i in progenitores.ravel():
                # Progenitores selecionados aleatóriamente  #####
                novos_individuos.append(pop[i])
                if verbose: print("Progenitores", pop[i])
                else: print('.',sep='',end='')
            for el in range(2):
                # Elite garantindo os 2 melhores da geração anterior
                bol, pos = busca_item_lista(elite[el],novos_individuos)
                if not bol:
                    if verbose: print('\t\t VIP', elite[el])
                    else: print('.',sep='',end='')
                    novos_individuos.append(elite[el])
            pop = novos_individuos
            novos_individuos = []
```

```
        list_t_pop.append(len(pop))
        #print('Novos Individuos:',novos_individuos)
        probabilidades = []
        if verbose: print('-->>>Aptidão-----------------------------------
-------------------')
        else: print('.',sep='',end='')
        for id, individuo in enumerate(pop):
            if verbose: print(ep,'#', id, individuo)
            else: print('.',sep='',end='')
            #bag, ibag = busca_item_lista(individuo, arca_gen)

            if tuple(individuo) in dic_arca:
                if verbose:
                    print('\t__Individuo Clone Recuperado da Arca #id:',
                        dic_arca[tuple(individuo)][0])
                else: print('.',sep='',end='')
                a = dic_arca[tuple(individuo)][3]
                m = dic_arca[tuple(individuo)][2]

                if verbose:
                    print('\t arca_apt recuperado:',a)
                    print('\t arca_mape recuperado:',m)
                else: print('.',sep='',end='')
                count_pop_r +=1
            else:
                m,a,yp = func_aptidao(individuo)
                dic_arca[tuple(individuo)]=[len(dic_arca),individuo,m,a]
                dic_arca_i[len(dic_arca_i)]=individuo
                arca_gen.append(individuo)
                arca_apt.append(a)
                arca_mape.append(m)
                count_pop_c +=1
            #print(probabilidades)
            if m>1:
                probabilidades.append(0)
            else:
                probabilidades.append(1-m)
            gera_mape.append(m)
            #print('\t\t individuo mape',individuo,m)
            #print('\t\t',gera_mape)
        list_t_pop_c.append(count_pop_c)
        list_t_pop_r.append(count_pop_r)
        probabilidades =(np.array(probabilidades)/sum(probabilidades))
        #print('\t população avaliada:', pop)
        if verbose:
            print('\t Probabilidades:', np.round(probabilidades,4))
            print('-->>>Elite-----------------------------------------------
-----------')
        else: print('.',sep='',end='')

        prob_elite = np.sort(probabilidades).tolist()[-2:]
        elite = [   pop[probabilidades.tolist().index(prob_elite[0])],
                    pop[probabilidades.tolist().index(prob_elite[1])]]

        if verbose:
            print('\t\t Elite :', elite)

            print('-->>>Progenitores-----------------------------------
-----------')
        else: print('.',sep='',end='')
        progenitores = selecao_de_progenitores(pop, probabilidades, casais)

        if verbose:
```

```
            print('-->>>Cruzamento-----------------------------------------
-----------')
        for p in progenitores:
            f1, f2 = cruzamento(p,pop, n_genes)
            novos_individuos.append(f1)
            novos_individuos.append(f2)
        if verbose:
            print('Novos Individuos F1:',f1)
            print('Novos Individuos F2:',f2)
        else: print('.',sep='',end='')
        #print('Novos Individuos:',novos_individuos)
        if verbose:
            print('-->>>Mutação-------------------------------------------
-----------')
        # Replicas com Mutação dos filhos
        mutantes = []
        for en, ori in enumerate(novos_individuos):
            mu = mutacao(ori,indice_mut,n_genes)
            mutantes.append(mu)
        novos_individuos += mutantes
        # Replicas com mutação da elite
        mutantes2 = []
        if verbose:
            print('Novos Individuos:',novos_individuos)
        else: print('.',sep='',end='')

        for en, ori in enumerate(elite):
            mu = mutacao(ori,indice_mut,n_genes)
            mutantes2.append(mu)
        novos_individuos += mutantes2
        if verbose: print('Novos Individuos:',novos_individuos)
        else: print('.',sep='',end='')
        ## Resultados Parciais ------------------------------------------
--------
        teste_mape += gera_mape
        ge_mape_min = (min(gera_mape))              # Menor erro na geração

        if verbose: print('\n#', ep, ge_mape_min)
        else: print('_#', ep, '_min MAPE',ge_mape_min,sep='',end='')
        res.append(ge_mape_min)
        ge_mape_ind = arca_mape.index(ge_mape_min)
        ge_mape_gen = arca_gen[ge_mape_ind]


        list_mutacao.append(indice_mut)             # Indice de proba
Mutação
        list_tdiff.append(time_diff(T1))            # Tempo de
Processamento

        ar_mape_min = min(arca_mape)     # Menor erro histórico
        ar_mape_ind = arca_mape.index(ar_mape_min)
        ar_mape_gen = arca_gen[ar_mape_ind]

        if verbose: relatorio_parcial()
        else: print('.',sep='',end='')

        #Salva resultados Parciais
        idtest = n_genes
        salva_resultado(ep,res,list_mutacao,list_tdiff,list_t_pop,
                        list_t_pop_r,list_t_pop_c,idtest)

        if bool_salva_arca: salva_arca(ep,idtest)
```

```
        # Critérios de parada
        if ge_mape_min < alvo:
            print('para por atingimento do alvo')
            break
        if verbose: print(f'\t Min Erro Atual {ge_mape_min:.5f} x
{min_anterior:.5f} Min Erro Anterior')
        else: print('.',sep='',end='')
        if round(ge_mape_min,5) == round(min_anterior,5):
            count_parada +=1
            if count_parada > 4:
                if indice_mut < 0.81:
                    indice_mut += 0.1
                    if verbose: print('Indice Mutação: ', indice_mut)
                    else: print('.',sep='',end='')
                    count_parada = 0
            if indice_mut > 0.81:
                indice_mut = 0.8

            if count_parada > max_parado:
                print('para por atingimento do limite de iterações parado')
                break
        else:

            min_anterior = min(ge_mape_min, min_anterior) #P/ Prox Laço
            count_parada = 0
        if verbose: print(f'\t parado a {count_parada} epocas')
        else: print('.',sep='',end='')

        if ep+1 == epocas:

            relatorio_parcial(ep,
                      count_parada,
                      arca_gen,
                      ar_mape_gen,
                      ar_mape_min,
                      ar_mape_ind,

                      ge_mape_gen,
                      ge_mape_min,
                      ge_mape_ind,

                      list_tdiff,
                      list_t_pop,
                      list_t_pop_c,
                      list_t_pop_r,
                      list_mutacao,
                      max_comb,
                      res,
                      teste_mape
                      )
    print('Parada por maximo de interações: ', ep+1              )
    print('Menor Mape  '                     , min(arca_mape)   )
    print('Código  '                         , ar_mape_gen      )
    print('Indice Mutação Final: '           , indice_mut       )

    atualiza_arca(bool_salva_arca)

    return  ar_mape_gen, min(arca_mape)

"""# Main (Código Principal)"""

atualiza_arca(False)
```

```
# Commented out IPython magic to ensure Python compatibility.
# %%time
# y_pred_list = []
# mape_list = []
# gen_list = []
#
# for e,i in enumerate(Ycolumns):
#     if 1:
#         shape_in, shape_out, X_train, y_train, X_test, y_test, y_real,
input_scaler, output_scaler = TrainTestData(e)
#         print('#'*10, e, i, '#'*20)
#         gen, mape = algoritmo_genetico(epocas=32, pop_tamanho=16)
#         m_laco, _, y_laco = func_aptidao(gen)
#         y_pred_list.append(y_laco)
#         gen_list.append(gen)
#         mape_list.append(m_laco)
# mape_list.append(np.array(mape_list).mean())

"""# roda

# Comparativo Validação
"""

#Relatório Manual agrupando os  resultados obtidos para cada SetorN1
if 0:
    list_mape = []
    list_gen = []

    ########## 0 Consumo        ####################
    list_mape.append(0.035652)
    list_gen.append(np.array([34, 58, 58, 14, 13,  7, 15,  1, 25, 26, 19,
37, 35]))
    ########## 1 Comercial      ####################
    list_mape.append(0.088187)
    list_gen.append(np.array([29,  9, 13, 10, 58, 58, 13, 15, 59,  3, 50,
28,  3]))
    ########## 2 Industrial     ####################
    list_mape.append(0.049014)
    list_gen.append(np.array([38, 19, 30,  7, 45, 41, 38, 58, 55,  6,  0,
25, 33]))
    ########## 3 Residencial    ####################
    list_mape.append(0.014144)
    list_gen.append(np.array([20, 26, 49, 28, 31, 15,  0, 48, 58, 52, 23,
19, 11]))
    ########## 4 Outros         ####################
    list_mape.append(0.024953)
    list_gen.append(np.array([57, 34,  1, 35, 59, 12, 57, 18, 48, 37, 39,
54, 60]))

    list_mape.append(np.array(list_mape).mean())
    barras = [*Ycolumns, 'Média']

    plt_barras(barras,list_mape)

Ycolumns

y_pred_list = []
mape_list = []

for e,i in enumerate(Ycolumns):
    if 1:
        shape_in, shape_out, X_train, y_train, X_test, y_test, y_real,
input_scaler, output_scaler = TrainTestData(e)
```

```
        print('\n'+('#'*10), e, i, ' '*(20-len(i))+'#'*20)
        m_laco, _, y_laco = func_aptidao(gen_list[e])
        y_pred_list.append(y_laco)
        mape_list.append(m_laco)
mape_list.append(np.array(mape_list).mean())

X,y,input_scaler,output_scaler,X_train,y_train,X_test,y_test,y_real,shape_i
n, shape_out = TrainTestData_old()

y_pred_list = np.array(y_pred_list).reshape(5,12)

plt_perfis_de_consumo2(y_pred_list.T)

"""#Fim"""

#def plt_perfis_de_consumo2(yp = y_pred_list.T):
if 1:
    yp = y_pred_list.T
    list_mape = []

    fig, axs = plt.subplots(2,3, figsize=(12,15))
    for e, i in enumerate(y.columns):
        if e < 3:
            s = 0
        elif e < 6:
            s = 1
        else:
            s = 2
        print(e, i)
        sns.lineplot(ax=axs[s,e-s*3],x=range(1,13),y=yp[-12:,e]/1000, label
= 'pred')
        sns.lineplot(ax=axs[s,e-s*3],x=range(1,13),y=y.iloc[-12:,e]/1000,
label = 'real')
        axs[s,e-s*3].set_title(i)
        axs[s,e-s*3].set_xlabel('')
        axs[s,e-s*3].set_ylabel('')
        #axs[s,e-s*3].set_ylim(0,3_000_000)
        axs[s,e-s*3].set_ylim(0)
        axs[s,e-s*3].set_xticks(range(0,13,2))
        axs[s,e-s*3].set_xlim(1, 12)
        axs[s,e-
s*3].yaxis.set_major_formatter(ticker.FuncFormatter(formatador_de_milhares)
)
        mape = mean_absolute_percentage_error(y.iloc[-12:,e],yp[:,e])
        list_mape.append(mape)
    list_mape.append(np.array(list_mape).mean())

    plt.suptitle('Consumo [kW] por SetorN1' , y=.93, fontsize=17)
    y_bars = [*y.columns,'Média']
    g = sns.barplot(ax=axs[1,2], x=y_bars, y=list_mape, palette='rocket')
    for i in range(len(y_bars)):
        g.text(i,list_mape[i]+0.001, round(list_mape[i],3), color='black',
ha="center", fontsize=10)
    plt.xticks(rotation = 90, fontsize=10)
    plt.title('MAPE por SetorN1', fontsize = 10)

    plt.show()
    ;
```