PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Emanuel Brandão de Galvão Correia

ANÁLISE DE SÉRIES TEMPORAIS E *FORECASTING* PARA O PREÇO MÉDIO DE COMBUSTÍVEIS DO ESTADO DE MINAS GERAIS EM PERÍODO PÓS PANDEMIA

Belo Horizonte
2024

Emanuel Brandão de Galvão Correia

ANÁLISE DE SÉRIES TEMPORAIS E *FORECASTING* PARA O PREÇO MÉDIO DE COMBUSTÍVEIS DO ESTADO DE MINAS GERAIS EM PERÍODO PÓS PANDEMIA

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

Belo Horizonte

2024

SUMÁRIO

1. Introdução

1.1. Contextualização

A pandemia da COVID-19, decretada oficialmente pela OMS em janeiro de 2020, foi um evento global que impactou a sociedade de uma forma totalmente nova, fazendo com que hábitos que antes achávamos ser banais precisassem ser revistos e adaptados para esta nova realidade. O impacto no cotidiano das pessoas foi enorme, mas sobretudo a sociedade sofreu consequências que levaram (ou levam) tempo para serem superadas. Setores como Saúde, Educação e Comércio emergiram como os principais afetados pela nova realidade, que logo exigiu adaptação e medidas para conter os danos sofridos.

Dentre os produtos que sofreram com a condição de distanciamento social e *lock-down* no Brasil há de se destacar o setor de combustíveis, bem extensamente utilizado por diversos setores e serviços da sociedade, que devido a restrição de mobilidade imposta pela situação global teve uma queda vertiginosa na sua demanda e consequentemente nos seus preços [1].

O setor de combustíveis constitui parte essencial do ecossistema econômico brasileiro, visto que está ligado aos serviços de transporte rodoviário e aeroviário de mercadorias e pessoas, setor alimentício e até investimentos. As flutuações ocorridas no preço deste recurso podem afetar desde o preço do alimento que chega na mesa do brasileiro até índices atrelados à inflação, como o IPCA [2].

Dessa forma, torna-se extremamente valiosa a análise histórica do preço dos combustíveis no Brasil, não apenas para avaliar como os diferentes eventos externos podem afetar a variação do preço dele, mas também para fornecer subsídio aos setores que necessitam incluir em suas receitas e previsões o valor médio ou aproximado dos diversos combustíveis que consome, seja mensalmente, semestralmente ou anualmente.

Este trabalho busca realizar uma análise de séries temporais do preço médio dos principais combustíveis (Etanol e Gasolina) nos períodos considerados como pós-pandemia (2022-atual)[1] no estado de Minas Gerais.

[1] Oficialmente a pandemia da COVID-19 foi decretada como finalizada pela OMS em 5 de maio de 2023. Apesar disso, neste trabalho a análise se iniciará a partir do segundo semestre de 2022, na qual os reflexos sofridos pelo setor analisado já estavam em fase de recuperação.

1.2. O problema proposto

O registro e acompanhamento da variação do preço de revenda dos combustíveis (praticado pelos postos ao consumidor) constitui ação essencial não só para o cidadão comum brasileiro, mas também, como já dito, para os diversos setores de serviços, economia e comércio da sociedade.

A imprevisibilidade do valor de venda deste bem, devido as diversas flutuações possíveis, pode ser um problema para o balanço financeiro de empresas, de pessoas físicas e de setores inteiros, como o de transporte por exemplo. Prever qual será o valor da Gasolina ou do Etanol em um período futuro é uma ferramenta muito útil que pode gerar vantagens comerciais, economia de custos e fornecer subsídios para tomada de ações que visem minimizar os danos financeiros e socioeconômicos dos diversos consumidores de combustíveis.

Este trabalho visa fornecer uma análise histórica e detalhada acerca da flutuação do preço da Gasolina e Etanol no estado de Minas Gerais após o período de recuperação econômica e social da pandemia global da COVID-19 no Brasil, englobando o período de 2022 até os dias atuais. Nesta análise buscaremos entender como o preço da revenda deste bem para o consumidor pode sofrer variações devido a fatores já bem conhecidos como mudança de governo, flutuações do valor do dólar e períodos do ano. Mas também estudar quais efeitos a pandemia trouxe a estes valores no contexto de recuperação após o período em que ela mais afetou o cenário econômico e social nacional.

A seguir, empregando técnicas de estatística e de *Machine Learning*, buscaremos realizar previsões (*Forecasting)* acerca do comportamento do preço médio dos dois principais combustíveis comercializados em MG em um período futuro definido. A fim de viabilizar análises mais profundas e tornar mais confidentes as decisões de consumidores que necessitem destas informações.

As informações e dados utilizadas no projeto foram fornecidos pela Agência Nacional do Petróleo, Gás Natural e Biocombustíveis (ANP) e disponibilizados no Portal de Dados Abertos do Governo Federal em forma de *datasets* contendo as séries históricas dos preços

de combustíveis em todo o Brasil. Este trabalho foi restringido a considerar o escopo das informações referentes aos anos de 2022, 2023 e 2024.

Os dados foram disponibilizados em cumprimento às determinações da Lei do Petróleo (Lei nº 9478/1997, artigo 8º), que institui à ANP o acompanhamento dos preços praticados por revendedores de combustíveis automotivos e de gás liquefeito de petróleo envasilhado em botijões de 13 quilos (GLP P13). Este acompanhamento é então feito por meio de uma pesquisa semanal de preços realizada por empresa contratada [3]. Este processo é o responsável pelo fornecimento dos dados deste projeto.

É possível sintetizar o problema e o objetivo macro do projeto ao utilizar a técnica dos 5-Ws, que visa elencar as principais perguntas que devem ser feitas na investigação de alguma situação ou problema, sendo aplicável a diversos cenários profissionais e de pesquisa.

Why? (Por quê?): Realizar análises de séries históricas de preços de produtos e previsões do valor em períodos futuros é importante pois fornece subsídio para tomadas de decisão de empresas e pessoas que dependem das variações do valor deste bem, fornecendo vantagens comerciais, redução de custos e iniciativas que visem evitar danos. Além disso é uma forma de estudar os diversos agentes que podem afetar as flutuações do valor de venda dos combustíveis, como fatores políticos, econômicos e/ou sociais.

Who? (Quem?): Os dados que serão analisados são de posse da Agência Nacional do Petróleo, Gás Natural e Biocombustíveis (ANP), sendo disponibilizados no Portal de Dados Abertos do Governo Federal do Brasil (gov.br).

What? (O quê?): O problema proposto é a análise de dados históricos do preço dos combustíveis no período pós-pandemia (2022-atual) em Minas Gerais e os efeitos dela no valor deste bem e nos seus valores futuros.

Where? (Onde?): O escopo geográfico do trabalho é restrito ao âmbito do estado de Minas Gerais.

When? (Quando?): O período a ser analisado compete a janeiro de 2022 até maio de 2024.

1.3. Versão do ambiente e bibliotecas utilizadas

Este trabalho foi realizado utilizando da linguagem de programação *Python*, na versão 3.11.5 (figura 1), executada no ambiente de desenvolvimento *Jupyter Lab*, que foi utilizado na sua versão 3.6.3 (figura 2).

Figura 1 – Versão da linguagem *Python*

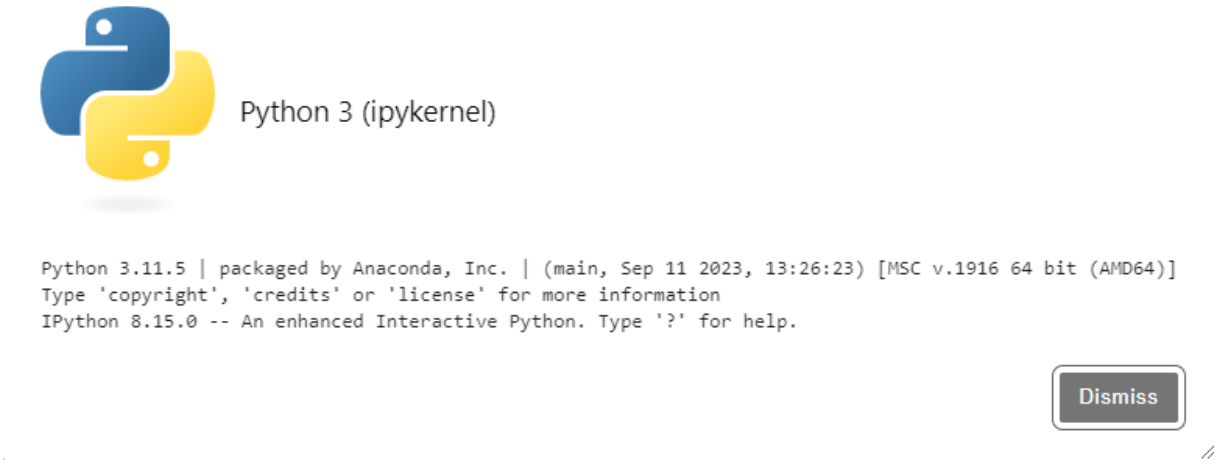
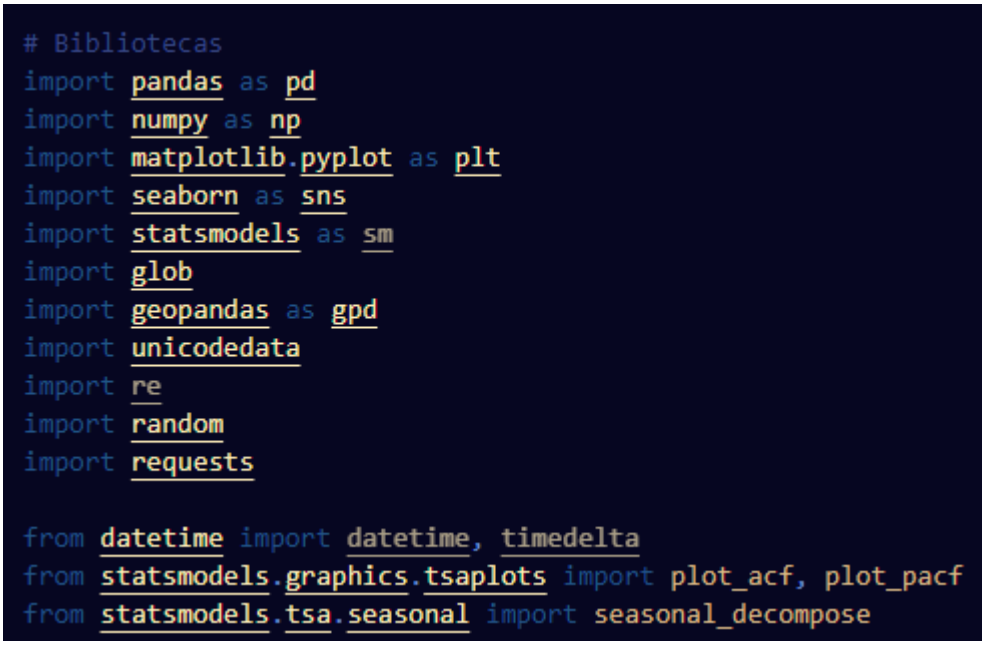


Figura 2 – Versão do *Jupyter Lab* utilizada

Além dos ambientes descritos acima, também foram importadas diversas bibliotecas ao ambiente, afim de fornecer subsídio para as principais etapas do projeto: Importação dos dados, preparação dos dados, análise dos dados, modelagem dos dados e visualização.

Dentre as bibliotecas, além das nativas do Python, destacam-se as bibliotecas de dados como *pandas e numpy*, as de visualização como *matplotlib e seaborn* e as utilizadas para o processo de Machine Learning como o *Scikit-learn, XGBoost e auto-arima*. Todas as bibliotecas utilizadas estão representadas na figura abaixo.

Figura 3 – Bibliotecas importadas no ambiente de execução do projeto



```
### BIBLIOTECAS UTILIZADAS NA ETAPA DE MACHINE LEARNING

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import glob
import warnings
warnings.filterwarnings('ignore')

## PRÉ-PROCESSAMENTO DOS DADOS

from sklearn.model_selection import train_test_split, TimeSeriesSplit
from sklearn.metrics import mean_absolute_error, root_mean_squared_error
from mlforecast import MLForecast
from mlforecast.lag_transforms import RollingMean
from numba import njit

## RANDOM FOREST e XGBOOST
from sklearn.ensemble import RandomForestRegressor
import xgboost

## SARIMA
import statsmodels as sm
from pmdarima import auto_arima

## LSTM
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import EarlyStopping

# OTIMIZAÇÃO
import optuna
optuna.logging.set_verbosity(optuna.logging.WARNING)
```

2. Coleta de Dados

2.1. Bases de dados

O conjunto de dados que foi escolhido para o desenvolvimento deste trabalho foi obtido do portal de dados abertos do Governo Federal do Brasil, acessado em janeiro de

2024 via o seguinte link: https://dados.gov.br/dados/conjuntos-dados/serie-historica-de-precos-de-combustiveis-e-de-glp.

Esse conjunto de dados contém informações acerca do preço dos seguintes combustíveis entre o período de Jan/2022 à Mai/2024:

- Gasolina Comum;
- Gasolina Aditivada;
- Etanol;
- Diesel;
- Diesel S10;
- Gás Natural Veicular (GNV);

Além da informação do preço também constam nos dados informações de endereço do local do estabelecimento que efetivou a compra como Região, Estado, Município etc. Por fim há também informações sobre o estabelecimento em si como seu nome, CNPJ, e qual sua Bandeira.

Com o intuito de enriquecer a análise do objeto principal deste trabalho e fornecer mais subsídio à modelagem preditiva, foram selecionadas outras duas bases de dados. A primeira se trata de uma API de CEP do Brasil, que contém informações de endereços a partir da base de dados do IBGE [5], essa base será importante para trazer informações geográficas que podem ser pertinentes à previsão do preço de combustíveis como Latitude e Longitude. A outra se trata da base de dados histórica dos valores da cotação do Dólar (USD) em Real brasileiro (BRL), no mesmo período em que se encontram os dados. A escolha dessa base de dados se dá por motivos de forte correlação entre essas duas variáveis: o preço do combustível e o valor da cotação do Dólar mundial [4], sendo um fator que possui potencial para auxiliar na capacidade preditiva da análise proposta. Os dados foram obtidos da plataforma *Investing.com*, acessada em maio de 2024, via o seguinte link: https://br.investing.com/currencies/usd-brl-historical-data.

Nesta última base de dados estão contidos, para o mesmo período (Jan/2022 à Mai/2024), o valor de abertura, fechamento, máximo e mínimo da cotação do Dólar em cada dia, bem como a variação percentual do preço do mesmo em relação ao dia anterior.

A seguir está representado como uma tabela a descrição dos campos presentes nos dados bem como seu tipo e nome:

a. Série histórica do preço dos combustíveis

Nome da coluna/campo	Descrição	Tipo
Regiao - Sigla	Sigla da Região da revenda pesquisada	*String*
Estado – Sigla	Sigla da Unidade Federativa (UF) da revenda pesquisada	*String*
Municipio	Nome do município da revenda pesquisada	*String*
CNPJ da Revenda	Número do Cadastro Nacional de Pessoa Jurídica da revenda pesquisada	*String*
Nome da Rua	Nome do logradouro da revenda pesquisada	*String*
Numero Rua	Número do logradouro da revenda pesquisada	*Integer*
Complemento	Complemento do logradouro da revenda pesquisada	*String*
Bairro	Nome do bairro da revenda pesquisada	*String*
Cep	Número do Código do Endereço Postal (CEP) do logradouro da revenda pesquisada	*String*
Produto	Nome do combustível pesquisado	*String*
Data da Coleta	Data da coleta do(s) preço(s)	*Datetime*
Valor de Venda	Preço de venda ao consumidor final praticado pelo revendedor, na data da coleta	*Float*
Valor de Compra	Preço de distribuição (preço de	*Float*

	venda da distribuidora para o posto revendedor de combustível)	
Unidade de Medida	Unidade de Medida	*String*
Bandeira	Noma da Bandeira da revenda	*String*

b. API de CEP com base de dados do IBGE

Nome da coluna/campo	Descrição	Tipo
CEP	CEP fornecido na requisição	*String – Campo JSON*
address_type	Tipo de endereço do CEP (Rua, Quadra etc.)	*String – Campo JSON*
Address_name	Nome do endereço	*String – Campo JSON*
Address	Endereço completo	*String – Campo JSON*
District	Bairro/Distrito do endereço	*String – Campo JSON*
State	Estado do CEP	*String – Campo JSON*
City	Cidade do CEP	*String – Campo JSON*
Lat	Latitude do CEP	*String – Campo JSON*
Lng	Longitude do CEP	*String – Campo JSON*
Ddd	DDD telefônico do CEP	*String – Campo JSON*
City_ibge	Código da cidade do CEP na base do IBGE	*String – Campo JSON*

c. Preços históricos do Dólar americano em Real brasileiro

Nome da coluna/campo	Descrição	Tipo
Data	Data da coleta da cotação	*Datetime*
Último	Valor de fechamento da moeda no dia	*Float*
Abertura	Valor de abertura da moeda no dia	*Float*
Máxima	Valor máximo da moeda no dia	*Float*
Mínima	Valor mínimo da moeda no dia	*Float*

Vol.	Volume negociado (não aplicável)	*Float*
Var%	Variação percentual do valor do dia em relação ao do dia anterior	*String*

2.2. Importação dos *datasets* e sua leitura

Ambos os *datasets* foram baixados das suas plataformas já descritas na seção anterior e então importados para o projeto no *Python*. Quanto à base de preço de combustíveis, foram importados mais de um arquivo (alocados cada um em um dataframe – df), que foram combinados em um *dataframe* único (df_combs) através da função *concat()*. Já a base de dados da cotação do dólar foi atribuída a um único *dataframe* (df_dolar), como mostra a figura abaixo.

Figura 4 – Importação dos *datasets*

```
# Carregamento das bases de dados para o Pandas
path_combs = r'C:\\Users\Emanuel\Desktop\PUCMG\TCC\Datasets'
path_dolar = r'C:\Users\Emanuel\Desktop\PUCMG\TCC\USD_BRLdata.csv'

### PREÇOS DE COMBUSTÍVEIS

files = glob.glob(path_combs + '/*.csv')
many_dfs = [pd.read_csv(file, sep=';') for file in files]
df_combs = pd.concat(many_dfs, ignore_index=False)

### COTAÇÃO DO DÓLAR

df_dolar = pd.read_csv(path_dolar, sep=',')
```

As primeiras linhas dos *dataframes* foram obtidas com a função *head()* e podem vistas a seguir.

Figura 5 – Primeiras linhas dos *datasets* de combustível e valor do Dólar respectivamente

	Regiao - Sigla	Estado - Sigla	Municipio	Revenda	CNPJ da Revenda	Nome da Rua	Numero Rua	Complemento	Bairro	Cep	Produto	Data da Coleta	Valor de Venda	Valor de Compra	Unidade de Medida	Bandeira
0	N	AC	RIO BRANCO	AUTO POSTO AMAPA - EIRELI	00.529.581/0001-53	VIA CHICO MENDES	3570	NaN	AREAL	69906-119	GASOLINA	03/01/2022	6,99	NaN	R$ / litro	VIBRA ENERGIA
1	N	AC	RIO BRANCO	AUTO POSTO AMAPA - EIRELI	00.529.581/0001-53	VIA CHICO MENDES	3570	NaN	AREAL	69906-119	ETANOL	03/01/2022	5,99	NaN	R$ / litro	VIBRA ENERGIA
2	N	AC	RIO BRANCO	AUTO POSTO AMAPA - EIRELI	00.529.581/0001-53	VIA CHICO MENDES	3570	NaN	AREAL	69906-119	DIESEL	03/01/2022	6,09	NaN	R$ / litro	VIBRA ENERGIA
3	N	AC	RIO BRANCO	AUTO POSTO AMAPA - EIRELI	00.529.581/0001-53	VIA CHICO MENDES	3570	NaN	AREAL	69906-119	GASOLINA ADITIVADA	03/01/2022	7,05	NaN	R$ / litro	VIBRA ENERGIA
4	N	AC	RIO BRANCO	AUTO POSTO AMAPA - EIRELI	00.529.581/0001-53	VIA CHICO MENDES	3570	NaN	AREAL	69906-119	DIESEL S10	03/01/2022	6,12	NaN	R$ / litro	VIBRA ENERGIA

	Data	Último	Abertura	Máxima	Mínima	Vol.	Var%
0	28.12.2023	4,8521	4,8323	4,8714	4,8175	NaN	0,53%
1	27.12.2023	4,8267	4,8107	4,8408	4,8018	NaN	0,26%
2	26.12.2023	4,8140	4,8513	4,8614	4,8135	NaN	-0,92%
3	22.12.2023	4,8585	4,8865	4,8908	4,8481	NaN	-0,53%
4	21.12.2023	4,8844	4,9146	4,9146	4,8633	NaN	-0,65%

3. Processamento/Tratamento de Dados

3.1. Pré-processamento dos dados

Antes do início à análise estatística e exploratória dos nossos dados se faz necessário analisar como eles estão dispostos dentro do nosso problema. Isso significa realizar uma análise prévia de valores omissos, *outliers* e se os tipos de dados estão adequados ao nosso problema. Para a realização de tais análises e ajustes, caso necessário, utilizaremos da biblioteca *pandas*, própria para este tipo de escopo.

As funções *shape(), describe()* e o método *dtypes* da classe *Dataframe* são algumas das formas que foram utilizadas para obter informações gerais acerca dos dados de modo a subsidiar esta etapa. As informações acerca dos *dataframes* estão expressas na figura 6.

Figura 6 – Dimensões dos *dataframes,* tipos de dados das colunas e presença de valores nulos

```
Dados sobre combustíveis:
Quantidade de linhas: 2083811
Quantidade de colunas: 16


Regiao - Sigla        object
Estado - Sigla        object
Municipio             object
Revenda               object
CNPJ da Revenda       object
Nome da Rua           object
Numero Rua            object
Complemento           object
Bairro                object
Cep                   object
Produto               object
Data da Coleta        object
Valor de Venda        object
Valor de Compra      float64
Unidade de Medida     object
Bandeira              object
dtype: object
```

```
Dados sobre cotação do Dólar (USD):
Quantidade de linhas: 625
Quantidade de colunas: 7


Data        object
Último      object
Abertura    object
Máxima      object
Mínima      object
Vol.        object
Var%        object
dtype: object
```

```
--------------------------------
Quantidade de valores nulos em cada coluna - Dataset COMBUSTÍVEIS:

Regiao - Sigla              0
Estado - Sigla              0
Municipio                   0
Revenda                     0
CNPJ da Revenda             0
Nome da Rua                 0
Numero Rua                686
Complemento           1618721
Bairro                   4549
Cep                         0
Produto                     0
Data da Coleta              0
Valor de Venda              0
Valor de Compra       2083811
Unidade de Medida           0
Bandeira                    0
dtype: int64
```

```
Quantidade de valores nulos em cada coluna - Dataset DÓLAR:

Data          0
Último        0
Abertura      0
Máxima        0
Mínima        0
Vol.        353
Var%          0
dtype: int64
```

3.2. Tratamento dos dados

O *dataframe* que contém os dados sobre combustíveis apresenta informações históricas de todo o Brasil, daí a existência de mais de 1.8 milhões de registros, além de conter também todos os combustíveis comercializados nacionalmente (Gasolina, Diesel, Etanol e GNV). A fim de adequar os dados ao escopo desse trabalho, ou seja, apenas ao estado de Minas Gerais foram realizados filtros nos dados iniciais, através de condições que foram passadas via *pandas,* essas condições restringem o escopo dos dados apenas ao estado de MG e exclui os casos cujo combustível não é Etanol ou Gasolina, visto que não são de interesse desse estudo. Por fim, modificou-se o nome de algumas colunas no *dataframe* de combustíveis e da cotação do dólar para facilitar o processo de tratamento dos dados, análise e modelagem. Essas ações podem ser vistas na figura 7.

Figura 7 – Aplicação de máscaras no *dataframe* de combustíveis para obtenção apenas de dados de MG e que são de Gasolina/Etanol e mudança no nome de colunas

```
# Filtrando pelos casos apenas de MG e eliminando os casos que não são Gasolina/Etanol
c1 = df_combs['Estado - Sigla'] == 'MG'
c2 = df_combs['Produto'] == 'GASOLINA'
c3 = df_combs['Produto'] == 'ETANOL'
df_fuels = df_combs[c1 & (c2 | c3)]
```

```python
# Remoção de colunas indesejadas e mudança nos nomes das restantes - Dataset COMBUSTÍVEIS
non_relevant = ['Regiao - Sigla', 'Revenda', 'CNPJ da Revenda', 'Nome da Rua', 'Numero Rua', 'Complemento', 'Bairro', 'Valor de Compra', 'Unidade de Medida']
new_name = {'Estado - Sigla': 'estado',
            'Municipio': 'municipio',
            'Cep': 'cep',
            'Produto' : 'tipo_comb',
            'Data da Coleta' : 'data',
            'Valor de Venda' : 'preco',
            'Bandeira' : 'bandeira'}
df_fuels = df_fuels.rename(columns=new_name)\
    .drop(non_relevant, axis=1)

# Remoção de colunas indesejadas e mudança nos nomes das restantes - Dataset DÓLAR
non_relevant = ['Abertura', 'Mínima', 'Vol.', 'Máxima']
new_name = {'Data': 'data',
            'Último': 'ultimo_dolar',
            'Var%': 'variacao_dolar'}
df_dolar = df_dolar.rename(columns=new_name)\
    .drop(non_relevant, axis=1)
```

É possível notar que grande parte das colunas de ambos os *dataframes* estão como tipo *object,* que no Python representa um tipo que se adequa a quase que qualquer tipo de dado, mas que apesar dessa vantagem não se adequa bem à análise exploratória, criação de gráficos dentre outras ações que iremos realizar com os dados. Isso levanta a necessidade da conversão de algumas colunas dos *dataframes* para o tipo desejado, que é o que foi feito utilizando das funções *astype() e to_datetime().* Além disso, foram feitas outras padronizações como a substituição do caractere separador de decimais de vírgula (,) para ponto (.) e a remoção do hífen no campo de CEP.

Figura 8 – Conversão das colunas para a tipagem adequada

```python
# Conversão de tipagem dos dados das colunas - Dataset COMBUSTÍVEIS
cvcols = ['estado', 'municipio', 'cep', 'tipo_comb', 'bandeira']
df_fuels[cvcols] = df_fuels[cvcols].astype("string")
df_fuels['preco'] = df_fuels['preco'].str.replace(',', '.').astype("float")
df_fuels['data'] = pd.to_datetime(df_fuels['data'], format='%d/%m/%Y')
df_fuels['cep'] = df_fuels['cep'].str.replace('-', '')

# Conversão de tipagem dos dados das colunas - Dataset COMBUSTÍVEIS

df_dolar['ultimo_dolar'] = df_dolar['ultimo_dolar'].str.replace(',', '.').astype("float")
df_dolar['variacao_dolar'] = df_dolar['variacao_dolar'].str.replace(',', '.').str.replace('%', '').astype("float")
df_dolar['data'] = pd.to_datetime(df_dolar['data'], format='%d.%m.%Y')
```

```
Dataset COMBUSTÍVEIS

estado       string[python]
municipio    string[python]
cep          string[python]
tipo_comb    string[python]
data         datetime64[ns]
preco               float64
bandeira     string[python]
dtype: object

------------
Dataset DÓLAR

data        datetime64[ns]
ultimo             float64
variacao           float64
dtype: object
```

Um dos campos que precisavam ter a correta tipagem dos dados e estrutura é o campo CEP, visto que ele será utilizado como parâmetro na requisição feita à API-CEP descrita na seção de Coleta de Dados. A requisição à API é feita então utilizando a biblioteca *requests*, através de uma função criada para lidar com as possíveis respostas da API (sucesso ou falha), o único parâmetro solicitado pela API é o CEP, enquanto como resposta à requisição obtemos os dados de endereço daquele CEP, a partir da base de dados do IBGE, para o propósito do nosso problema iremos utilizar a latitude e longitude.

Figura 9 – Requisição à API-CEP e *merge* com o *dataframe* principal

```python
# CRIANDO COLUNA DE LAT E LONG A PARTIR DO CEP (VIA API)
## FUNÇÃO PARA OBTENÇÃO DA LAT/LONG VIA CEP - DOCS: https://docs.awesomeapi.com.br/api-cep

def get_lat_long(cep):
    req = requests.get(f"https://cep.awesomeapi.com.br/json/{cep}")
    if req.status_code == 200:
        lat = float(req.json()['lat'])
        lon = float(req.json()['lng'])
    else:
        lat = np.nan
        lon = np.nan

    return lat, lon

#-------------------------------------------------------------------------------#
## CRIAÇÃO DAS COLUNAS EM UM DF AUXILIAR
cep_to_conv = pd.DataFrame(df_fuels['cep'].unique(), columns=['cep'])
cep_to_conv[['lat', 'lon']] = cep_to_conv['cep'].apply(lambda x: pd.Series(get_lat_long(x)))
```

```python
# MERGING DOS DADOS DE LAT/LONG
df_fuels = df_fuels.merge(cep_to_conv[['cep', 'lat', 'lon']], on='cep', how='left')
```

É possível notar na figura acima que para os casos em que a requisição não retornar com Status 200 (Sucesso) o valor para a latitude/longitude será zero. Veremos que tais casos serão tratados mais tarde visto que existem CEPs na nossa base de dados que não constam na base de dados do IBGE. Além disso foi necessário criar um *dataframe* auxiliar contendo todos os CEPs únicos para reduzir o número de requisições à API, pois de acordo com a documentação dela, temos que para cada CEP é realizada uma requisição. Dessa forma foi possível reduzir o número de requisições de aproximadamente 100.000 (tamanho da base de dados *df_fuels*) para 1000 (CEPs únicos). Ao fim da obtenção dos dados da API foi realizado o *merge* com o *dataframe* contendo os dados de combustíveis.

A gasolina e o etanol, combustíveis objetos desse trabalho, possuem suas próprias características, como tendências, sazonalidades e variáveis diferentes que os influenciam. Devido a isso, para a análise exploratória dos dados, optou-se por separar o *dataframe* de combustíveis em *dataframes* para **cada** combustível afim de facilitar a análise, enquanto para a etapa de modelagem não será um problema utilizar do *dataframe* completo, visto

que basta tratar cada série temporal como um grupo dentro do modelo. A separação dos *dataframes* foi feita utilizando de um dicionário em Python.

Figura 10 – Criação de um dicionário contendo os *dataframes* para cada combustível

```
# CRIANDO 1 DF para cada combustível
gr = df_fuels.groupby('tipo_comb')
dfs = {}
for comb, data in gr:
    dfs['df_' + comb] = data


# RENOMEANDO os DFs
new_names = ['df_etanol', 'df_gasolina']
dfs = dict(zip(new_names, list(dfs.values())))
```

Dessa forma, é possível acessar cada *dataframe* chamando via chave do dicionário: *df_etanol e df_gasolina*.

Por fim, a última etapa do pré-processamento de dados consistiu em transformar os dados que continham múltiplas ocorrências do preço do combustível por dia, correspondente a cada posto que reportou os dados naquele dia, para uma ocorrência de preço por dia. Esse processo pode ser realizado criando variáveis **representativas** para cada dia, sendo essas transformações das variáveis já existentes adequadas a contemplarem apenas uma instância/ocorrência do preço do combustível por dia. Tal transformação é pautada no fato de que se deseja, no problema deste trabalho, prever o preço médio do combustível no dia, considerando influências conjuntas dos diversos estabelecimentos assinalados na base de dados tais como número de postos que reportaram o preço no dia, localizações médias destes postos etc.

As variáveis representativas criadas estão expressas na tabela abaixo, bem como o método utilizado para obtê-las:

a. Variáveis representativas para os *datasets* de combustíveis

Nome da variável original	Nome da variável representativa criada	Método utilizado
preco	**preco_medio**	Média
preco	preco_std	Desvio padrão
preco	preco_min	Mínimo
preco	preco_max	Máximo
cep	num_postos	Contagem
lat	lat_medio	Média
lon	lon_medio	Média

Figura 11 – Criação das variáveis representativas do dia e substituição das antigas

```
## CRIAÇÃO DE COLUNAS REPRESENTATIVAS DO DIA
for df in dfs:
    dfs[df]['preco_medio'] = dfs[df].groupby('data')['preco'].transform('mean')
    dfs[df]['preco_std'] = dfs[df].groupby('data')['preco'].transform('std')
    dfs[df]['preco_min'] = dfs[df].groupby('data')['preco'].transform('min')
    dfs[df]['preco_max'] = dfs[df].groupby('data')['preco'].transform('max')
    dfs[df]['num_postos'] = dfs[df].groupby('data')['cep'].transform('count')
    dfs[df]['lat_medio'] = dfs[df].groupby('data')['lat'].transform('mean')
    dfs[df]['lon_medio'] = dfs[df].groupby('data')['lon'].transform('mean')


for df in dfs:
    dfs[df] = dfs[df].drop(['estado', 'municipio', 'cep', 'bandeira', 'preco', 'lat', 'lon'], axis=1)
    dfs[df] = dfs[df].drop_duplicates(subset='data')
    dfs[df] = dfs[df].merge(df_dolar, on='data', how='left')
```

Com a obtenção de novas variáveis seja via API-CEP ou com a criação das colunas representativas fez-se necessário preencher as colunas que tinham valores *missing*, utilizando de um método coerente e adequado. Essa imputação fora realizada seguindo o que está expresso na tabela abaixo:

a. Colunas com imputação em valores *missing*

Nome da variável	Método utilizado
lat_medio	**Moda**
lon_medio	Moda

ultimo_dolar	Última ocorrência
variacao_dolar	Zeros
preco_std	Zeros
data	Última ocorrência

As variáveis geográficas (latitude/longitude) tiveram seus valores *missing* causados por CEPs não presentes na base do IBGE, devido a isso os valores foram imputados com as latitudes/longitudes médias mais recorrentes (moda), de modo que isso não trouxesse valores muito artificiais para a base. As variáveis do *dataset* do preço do dólar, *ultimo_dolar e variacao_dolar* tiverem respectivamente a utilização do método de última ocorrência, por se tratar de valores *missing* devido ao fim de semana e zero também pela variação do dólar não ocorrer nos dias que o valor constava como *missing*, este inclusive foi o mesmo caso da variável *preco_std*.

A variável temporal de data para cada *dataset* de combustível teve um *resample* para dias úteis, visando eliminar buracos de datas na base de dados.

Figura 12 – Imputação e resample das variáveis com valores *missing*

```
## PREENCHIMENTO DE COLUNAS COM VALORES MISSING
### mf = Most frequent

for df in dfs:
    dfs[df] = dfs[df].set_index('data')
    dfs[df] = dfs[df].resample('B').ffill()
    dfs[df].reset_index(inplace=True)

    mf_lat = dfs[df]['lat_medio'].mode()[0]
    mf_lon = dfs[df]['lon_medio'].mode()[0]

    dfs[df]['lat_medio'] = dfs[df]['lat_medio'].fillna(mf_lat)
    dfs[df]['lon_medio'] = dfs[df]['lon_medio'].fillna(mf_lon)
    dfs[df]['ultimo_dolar'] = dfs[df]['ultimo_dolar'].ffill()
    dfs[df]['variacao_dolar'] = dfs[df]['variacao_dolar'].fillna(0)
    dfs[df]['preco_std'] = dfs[df]['preco_std'].fillna(0)
```

4. Análise e Exploração dos Dados

Uma das etapas mais importantes do processo de Ciência de Dados ao abordar um problema, a EDA – *Exploratory data analysis* ou simplesmente Análise exploratória de dados deste problema foi dividida em três partes:

- Análise de variáveis categóricas;
- Análise de variáveis numéricas;
- Análise de séries temporais;

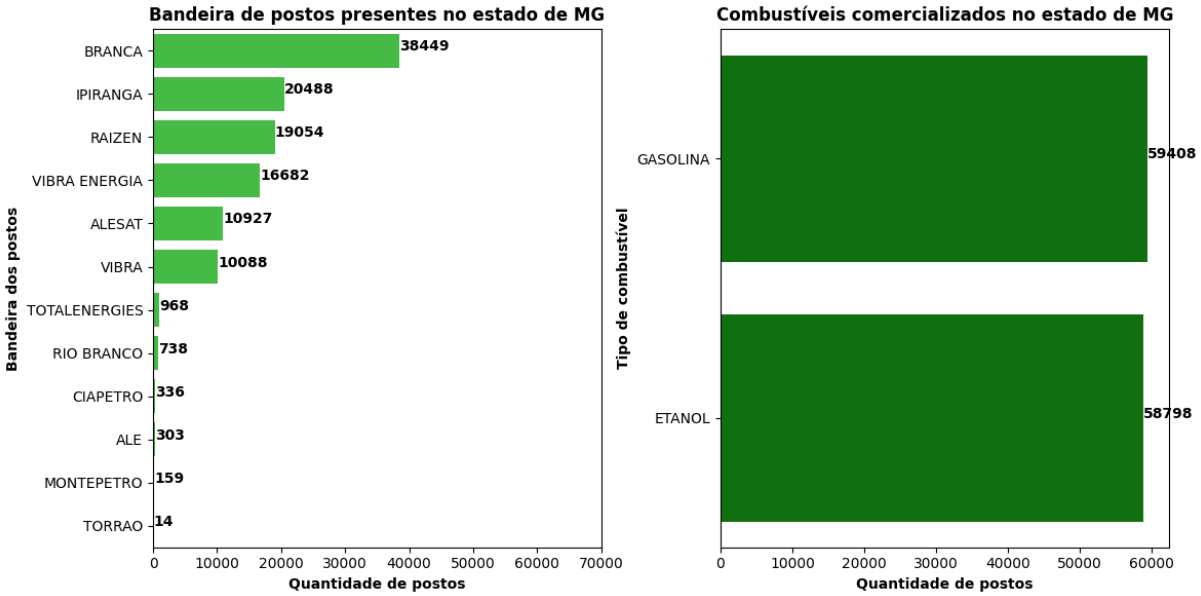
Em cada uma dessas partes foram exploradas as variáveis pertinentes ao problema e que foram mantidas na base de dados após a etapa de pré-processamento, bem como suas relações e influência com o que se deseja prever. No geral foram produzidos gráficos para auxiliar e subsidiar a análise dos dados, utilizando das bibliotecas de visualização *matplotlib e seaborn*.

4.1. Análise de variáveis categóricas

As variáveis categóricas presentes na base de dados de combustíveis foram as bandeiras dos postos que participaram da pesquisa semanal, além de respectivamente qual eram os combustíveis que tiveram seu preço registrado naquele posto, por fim havia também a variável município, representado a localidade do posto no estado de Minas Gerais.

A fim de analisar a proporção de ocorrência das variáveis citadas, foram criados dois gráficos de barras, o primeiro representando a contagem das bandeiras distintas dos postos da base de dados, e a segunda a ocorrência dos combustíveis gasolina e etanol.

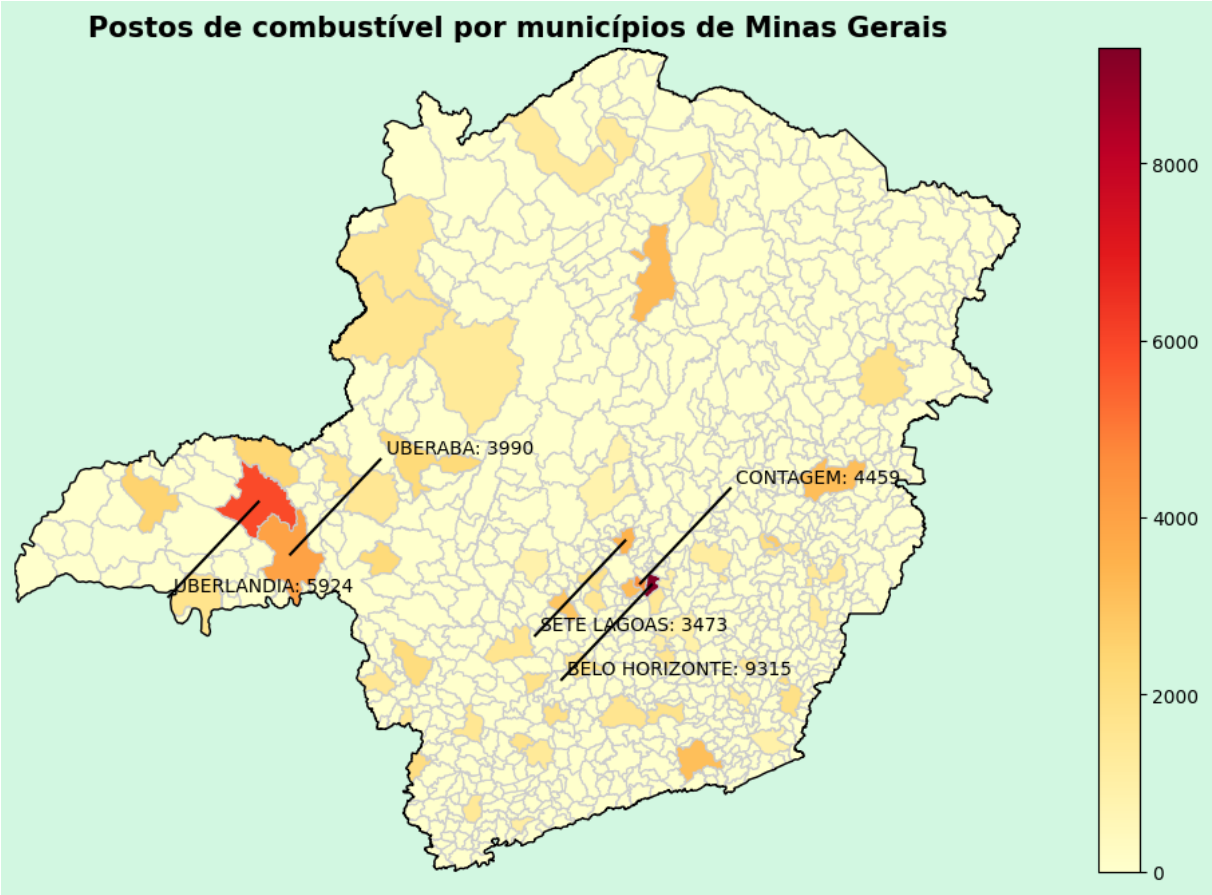
Figura 13 – Gráfico de barras da contagem das variáveis bandeira e combustível



A partir destes gráficos é possível notar uma proporção muito próxima da quantidade de postos que reportou os preços tanto da gasolina quanto do etanol, este fato é importante visto que invalida a existência de um erro na medição, o que poderia privilegiar alguma das bases de dados e inserir viés à nossa análise, dessa forma não se faz necessário realizar nenhum balanceamento nos dados.

Foi feita uma análise similar a variável município, na qual buscou-se também visualizar qual foi a ocorrência de cada município na base de dados, todavia por se tratar de uma variável também geográfica foi mais adequado fazer a visualização em formato de mapa.

Figura 14 – Mapa de calor dos municípios com mais postos que participaram da pesquisa de combustíveis



Representando o estado de Minas Gerais e seus municípios, o mapa acima foi construído usando a biblioteca *geopandas*, que espera um arquivo *geojson* contendo os respectivos estados/municípios bem como as coordenadas deles para terem seus limites traçados no gráfico. Neste trabalho foram necessários dois arquivos *geojson*, um para delimitar o estado de Minas Gerais e outro para delimitar os municípios, ambos os arquivos foram obtidos em repositórios do *GitHub* com fonte do IBGE [6][7].

No mapa é possível notar uma predominância de número de postos maior entre as cidades mais populosas do estado, como Belo Horizonte, Uberlândia e Contagem [8], além da concentração de postos em alguns municípios específicos. Isso pode ter origem das características não só regionais como o grande número de municípios, seu tamanho e população, mas também dos desafios inerentes à coleta de dados como logística e

dificuldade para uma abrangência homogênea no Estado. Devido a essas características é importante incluir a característica regional na modelagem dos preços dos combustíveis através de alguma variável, e para evitar tais problemas inerentes à pesquisa é que foram utilizadas a latitude e longitude obtidas a partir da API-CEP, como citado na etapa de processamento de dados, ao invés da variável município. A latitude e longitude, quando agregadas podem representar regionalidades geográficas, além de permitirem incluir municípios que tiveram poucos postos pesquisados na análise sem perder seu valor.

4.2. Análise de variáveis numéricas

Foco principal deste trabalho, o preço médio dos combustíveis etanol e gasolina é uma das variáveis numéricas a serem analisadas nesta etapa, bem como outra variações dela, tais como o preço mínimo e máximo naquele dia, seu desvio padrão e o preço médio do dólar no mesmo período, a fim de buscar relações entre estes e entender como está feita a distribuição dos dados.

A primeira visualização obtida foi um histograma contendo a distribuição dos valores de preço médio, para cada combustível, juntamente com a distribuição do desvio padrão deste preço em cada dia. Em cada histograma também foi estimado uma função densidade de probabilidade via método KDE (*Kernel Density Estimator*) [9]. Os histogramas de cada combustível juntamente com o dólar (para comparação) estão representados nas figuras abaixo.

Figura 15 – Histograma do preço médio, desvio padrão e dólar para o Etanol

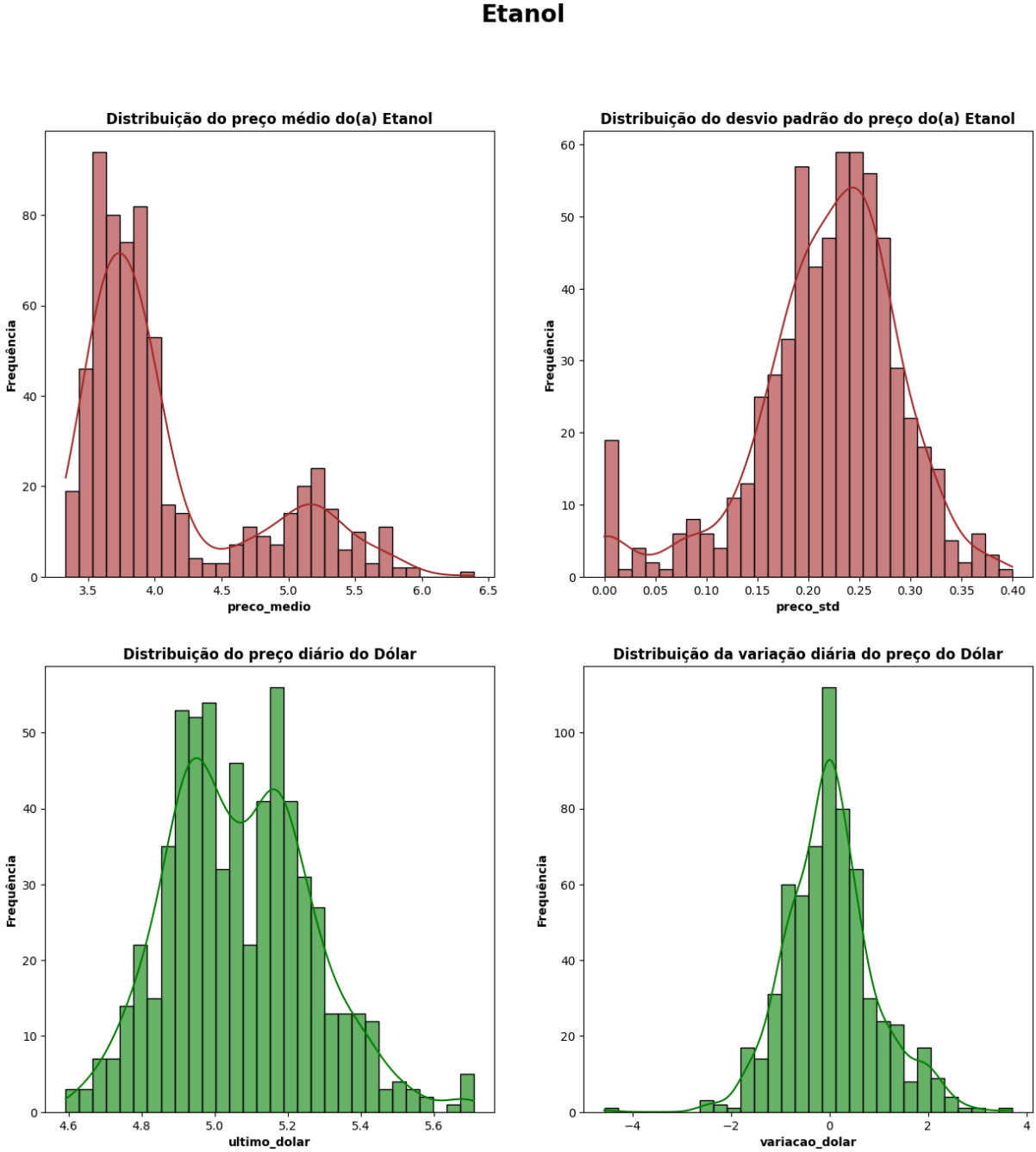
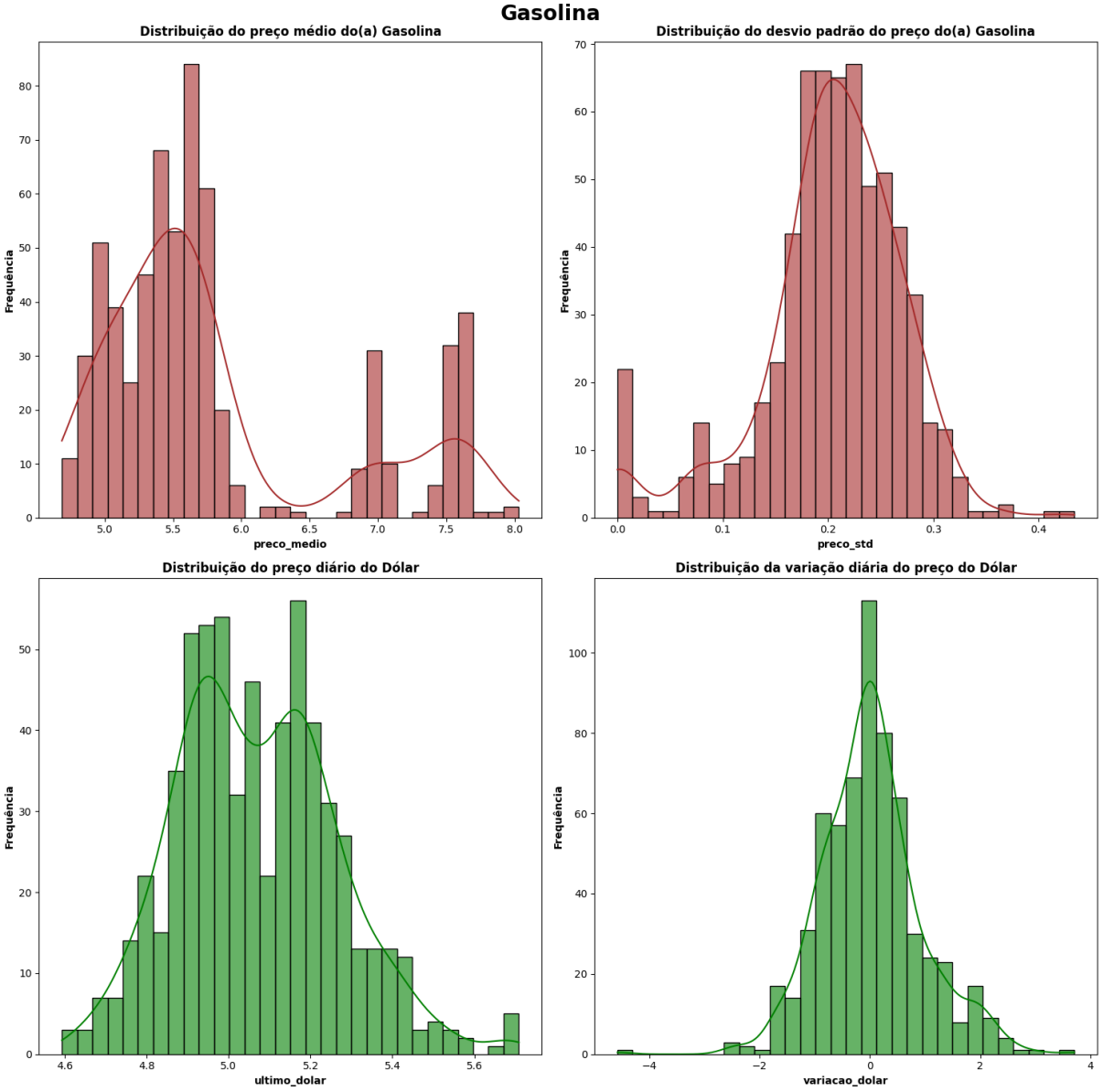


Figura 16 – Histograma do preço médio, desvio padrão e dólar para a Gasolina



Analisando o histograma do preço médio do **Etanol** é possível notar que a distribuição dos preços médios no período analisado têm uma maior concentração (e um maior pico) em torno de uma faixa de R$3,50 à R$4,00, todavia não é uma distribuição normal e sim assimétrica, visto que possui outro pico menor em torno da faixa de R$5,00 à R$5,50 indicando preços *outliers*, tal fator colabora na premissa de que variáveis externas são importantes para entender o comportamento desta variável, especialmente períodos significativos como o período pandêmico que buscamos estudar nesse trabalho.

A variabilidade do preço médio do Etanol, representada pelo histograma do desvio padrão, mostra que em geral o combustível sofre variações diárias em torno de R$0,15 à R$0,25, correspondentes ao maior pico do histograma, e que essa variabilidade é estável afinal o histograma se aproxima de uma distribuição normal. Isso revela um comportamento em que o combustível embora tenha dinamicidade diária no seu preço, não apresenta ocorrência de variações bruscas à curto prazo.

Uma análise semelhante à do Etanol pode ser aplicada aos *plots* da **Gasolina**, visto que ela apresenta, só que em uma faixa diferente de preço médio, dois picos maiores de ocorrência desta variável, todavia com preços *outliers* mais distantes da média (R$5,50), em torno da faixa de R$7,00 à R$7,50. Apesar de preços médios mais discrepantes, a variabilidade fornecida pelo desvio padrão é muito semelhante ao do Etanol, em torno de R$0,20, o que fortalece a premissa de que a variabilidade do preço ocorre, mas de forma estável, dessa forma os preços médios mais distantes da moda são obtidos à longo prazo e não de um dia para o outro.

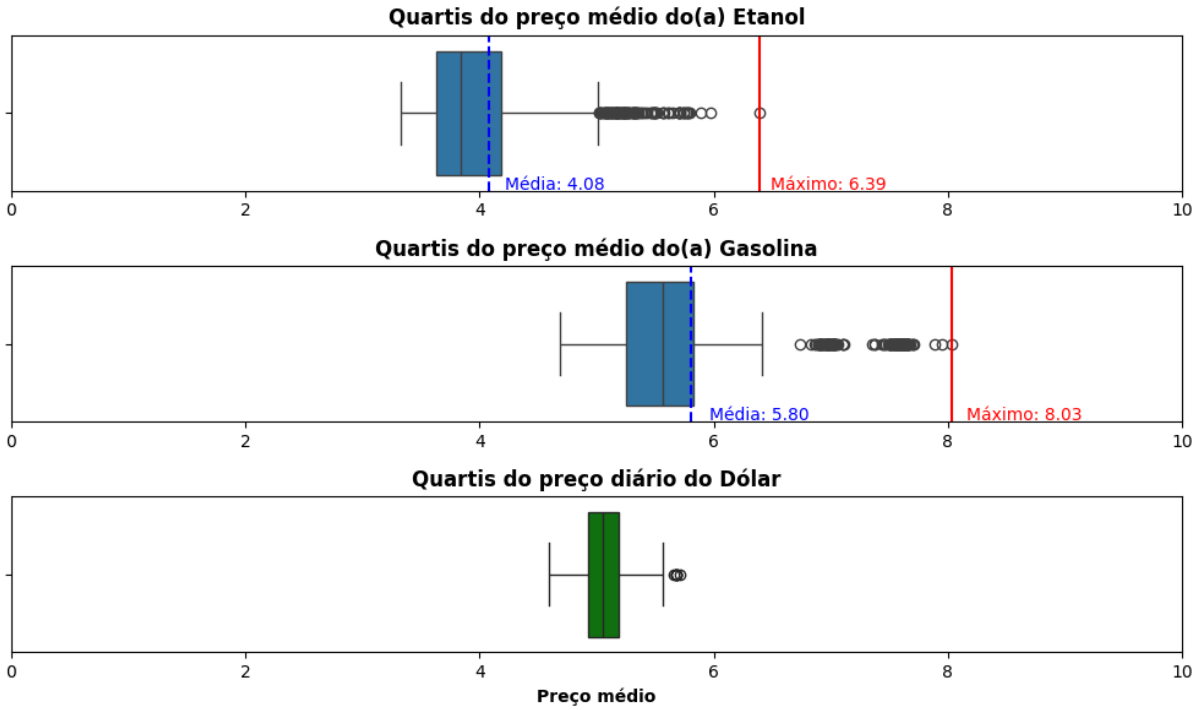
Em ambos os histogramas dos combustíveis, para fins de análise e busca de correlação com os preços deles, tivemos os histogramas do preço médio e variação do **dólar**, de acordo com a base de dados utilizada neste trabalho. Visualiza-se uma distribuição bimodal do preço médio da moeda, indicando que durante o período o preço flutuou entre dois preços de maior ocorrência, na faixa de R$4,80 e R$5,20. Isso pode ter relação com eventos que influenciam a mudança do preço da moeda, bem como variáveis macroeconômicas pertinentes a esse tipo de variável. Apesar disso, a presença de preços *outliers* aqui é bem menor que se comparada aos combustíveis.

Por fim, quanto à variabilidade do preço da moeda, é possível identificar uma distribuição normal centrada em zero, indicando pouca variabilidade diária, mas com mudanças esporádicas na moeda, diferentemente dos combustíveis, que sofrem ajuste diário constante, porém estável.

Os gráficos acima trazem então alguns fatores importantes para a nossa análise, como por exemplo a volatilidade diária do preço dos combustíveis, que ocorre de forma controlada, porém evidente, indicando constante mudança no valor da variável. Além disso há uma resposta conjunta dos preços do combustível à do dólar, visto que os histogramas de

preço médio de cada combustível e da moeda apresentam comportamentos semelhantes. Por fim, mas não menos importante, é significativa a presença de valores *outliers* para os preços médios dos combustíveis, fortalecendo a premissa de que variáveis externas podem estar relacionadas com esses valores discrepantes. Uma outra forma de visualizar a presença destes *outliers* é com o auxílio de um *boxplot*, o outro gráfico realizado na análise numérica desta etapa.

Figura 17 – *Boxplot* das variáveis de preço médio dos combustíveis e dólar



O *boxplot* ou diagrama de caixa é uma ferramenta de estatística descritiva que para o nosso trabalho fornece uma análise complementar aos histogramas apresentados anteriormente. Com os *boxplots* podemos entender melhor a presença dos valores *outliers* para a variável preço dos combustíveis e do dólar e visualizar a faixa dos valores dessas variáveis dentro do problema [10].

No diagrama dos combustíveis é possível notar uma significativa presença de valores fora da faixa interquartil, corroborando com as distribuições do preço deles. Especificamente para a gasolina é possível notar que esses *outliers* estão mais distantes do indicador do último quartil comparativamente ao etanol, o que indica uma volatilidade maior para o

preço da gasolina e consequentemente uma maior dispersão, algo que pode ser visto também com a maior assimetria do histograma dela.

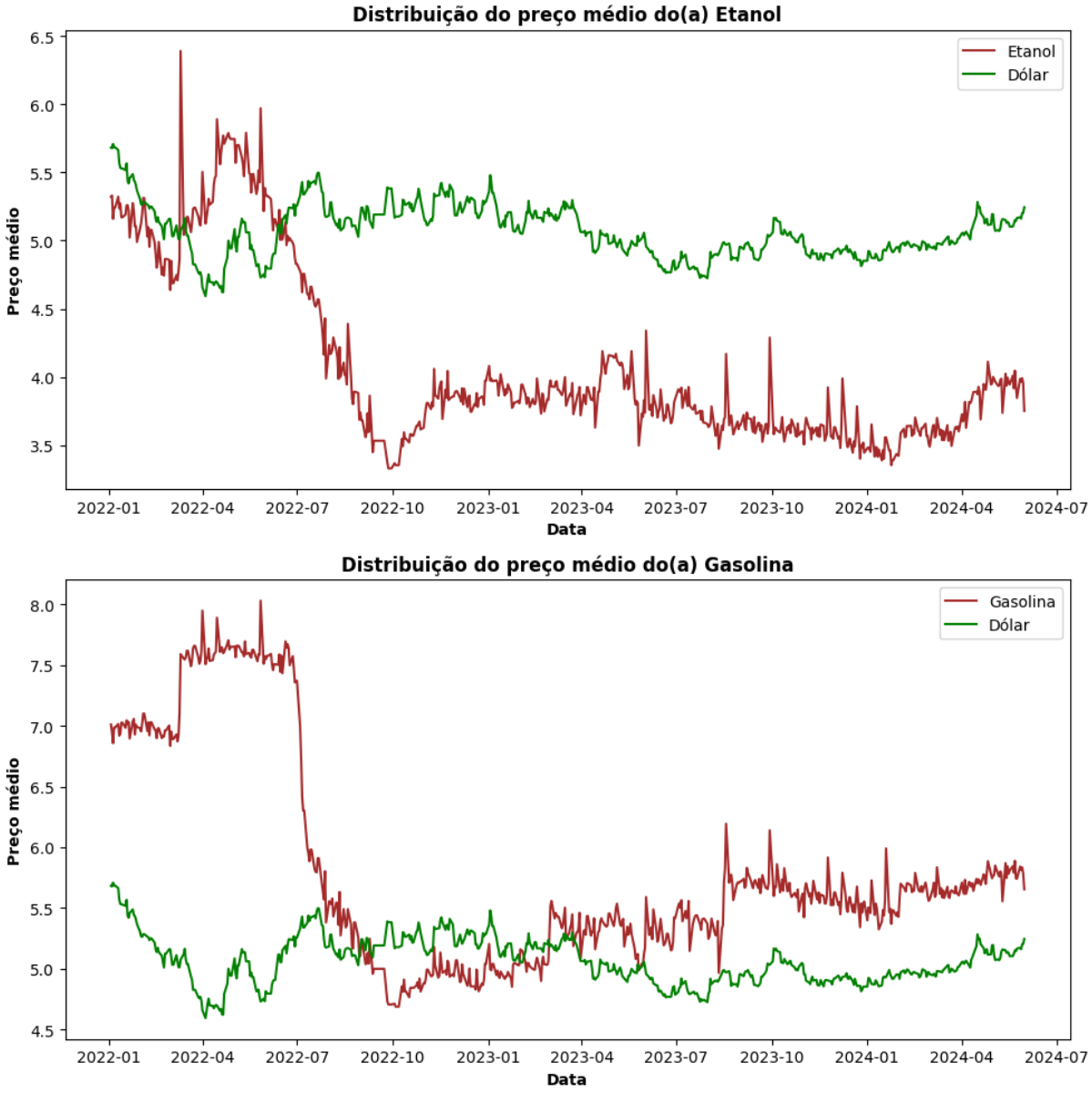
O dólar em comparação apresenta um *boxplot* mais enxuto, com distâncias interquartis menores e poucos *outliers*, colaborando com a normalidade na sua variação vista nos histogramas e fornecendo mais estabilidade ao seu preço no período analisado. O comparativo entre dólar e combustíveis é importante para entender que, o primeiro está mais sujeito a variações de nível global, enquanto o segundo é mais influenciado por variáveis nichadas/de setor, e por isso que o dólar é um termômetro interessante para utilizar como influência na variável combustível.

4.3. Análise de séries temporais

A última análise realizada nesta etapa de exploração de dados se refere a como as características temporais da base de dados e do problema podem afetar as distribuições e variações dos preços dos combustíveis. Na análise de séries temporais é importante buscar identificar componentes dela como tendência, sazonalidade e os resíduos dessa decomposição. Além de uma análise no tempo é importante entendermos se existe correlação entre todas as variáveis colocadas e a autocorrelação da variável temporal do problema, questão cerne em um problema de séries temporais e que é necessária de ser feita para subsidiar uma boa elaboração de modelo de previsão [11].

O primeiro recurso gráfico elaborado foi um *plot* das séries temporais dos preços médios dos combustíveis analisados, onde em ambos também foi incluída a série histórica do preço do dólar, para fins de comparação de comportamento e período.

Figura 18 – Gráficos de linha dos preços médios do Etanol e Gasolina em face do preço do dólar ao longo do tempo



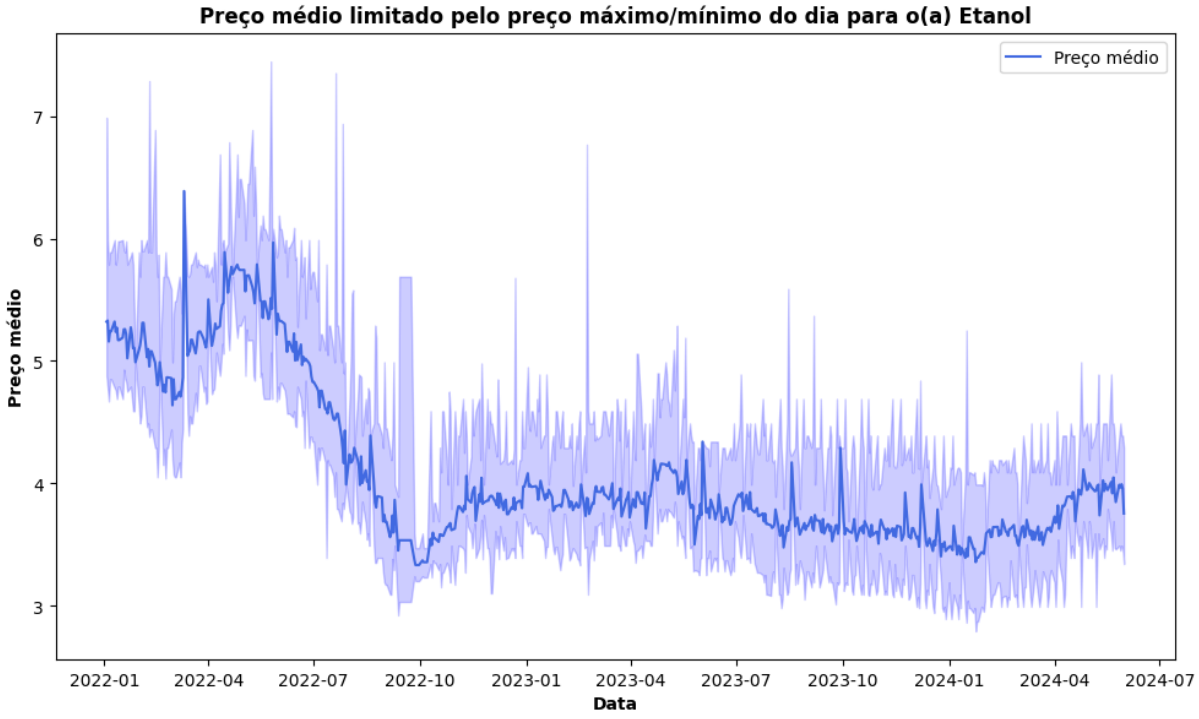
Os diagramas de séries temporais acima tanto para o Etanol quanto para a Gasolina revelam a grande característica do período pandêmico, sobretudo durante o primeiro semestre de 2022, o alto preço dos combustíveis comparado aos demais períodos nos anos seguintes. Foi também nesse período de 2022 que ambas as séries atingiram seus picos de valor médio máximo. Esse comportamento pode ser derivado das diversas variáveis externas que afetaram os combustíveis de forma mais incisiva nesse período, como a oferta e

demanda nos municípios do Estado, custo de produção e a disponibilidade. A partir do encaminhamento para o período de 2023 ocorre para ambos os combustíveis um declínio no preço, seguido de uma tendência mais estável até os últimos períodos analisados nesse trabalho.

A volatilidade do preço dos combustíveis não é observada no diagrama do preço do dólar, que se comparado em cada série, apresenta uma tendência mais estável em todo o período analisado, ficando alheio às altas variações que os combustíveis tiveram no primeiro e segundo semestre de 2022. Isso sugere uma relação mais indireta entre a variação do preço do dólar e dos combustíveis, aumentando ainda mais a premissa de que variáveis exógenas podem ser mais determinantes na variação do preço desses produtos do que somente a variação cambial de um dos possíveis lastros desses preços.

A fim de entender melhor as volatilidades dos preços dos combustíveis citadas acima, especialmente nos meses de maior ênfase da pandemia no período analisado no trabalho (01/2022 até 07/2022) é importante obtermos como diagramas auxiliares as mesmas séries temporais do preço médio, porém situadas em uma faixa de valores, cujos limites podem ser os preços máximo/mínimo do dia ou o desvio padrão.

Figura 19 – Série temporal do preço médio dos combustíveis com limites de preço mínimo e máximo no dia



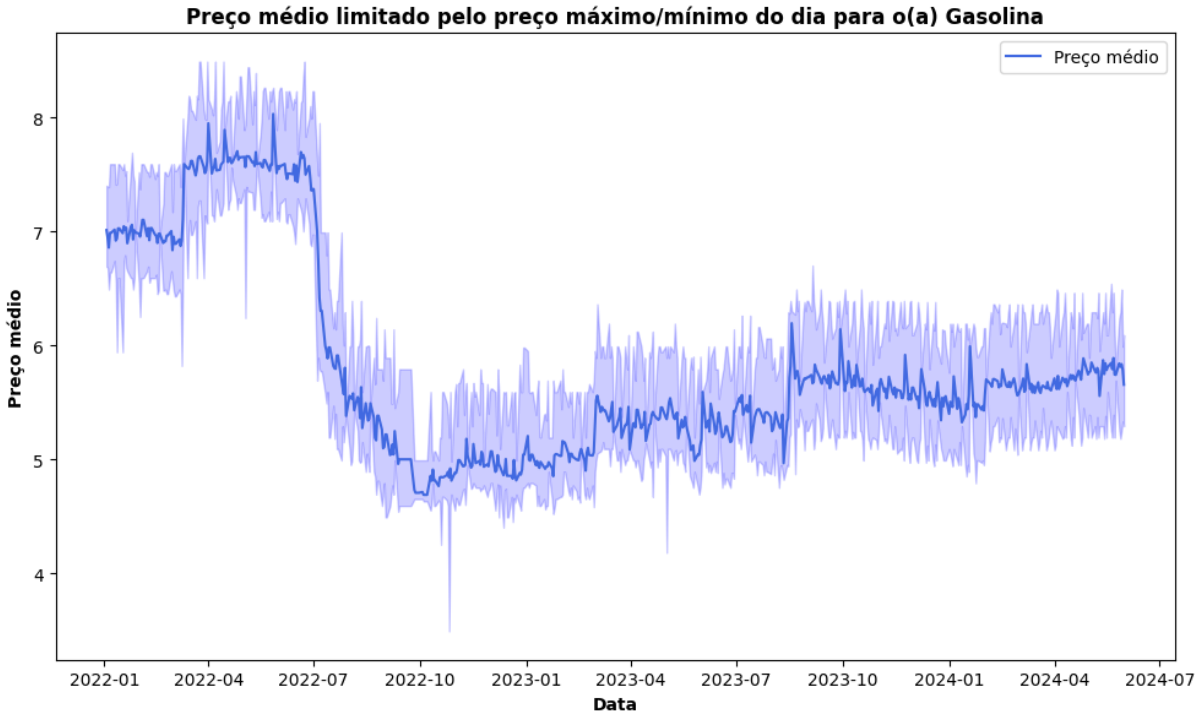
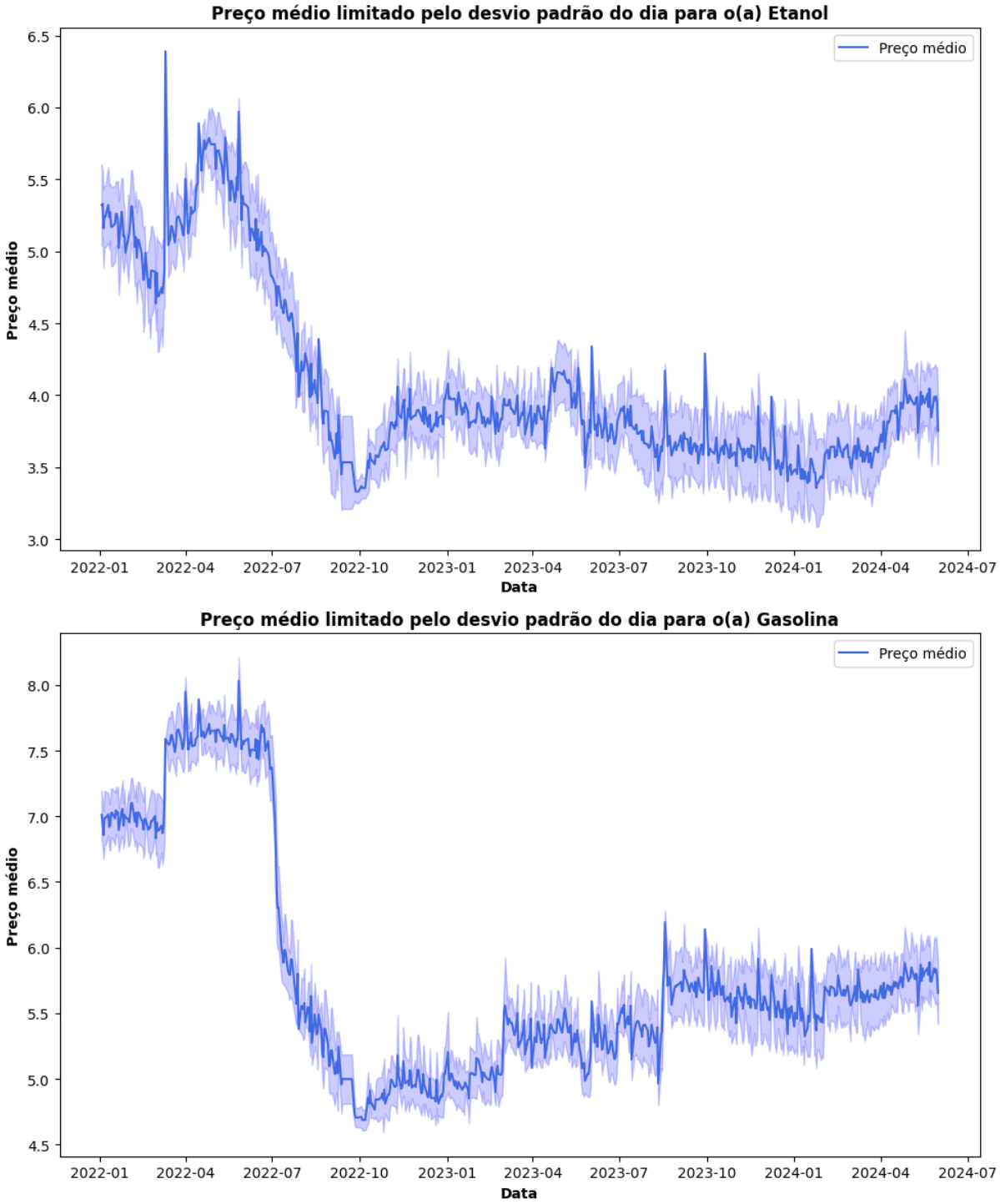


Figura 20 – Série temporal do preço médio dos combustíveis limitados pelo desvio padrão do dia



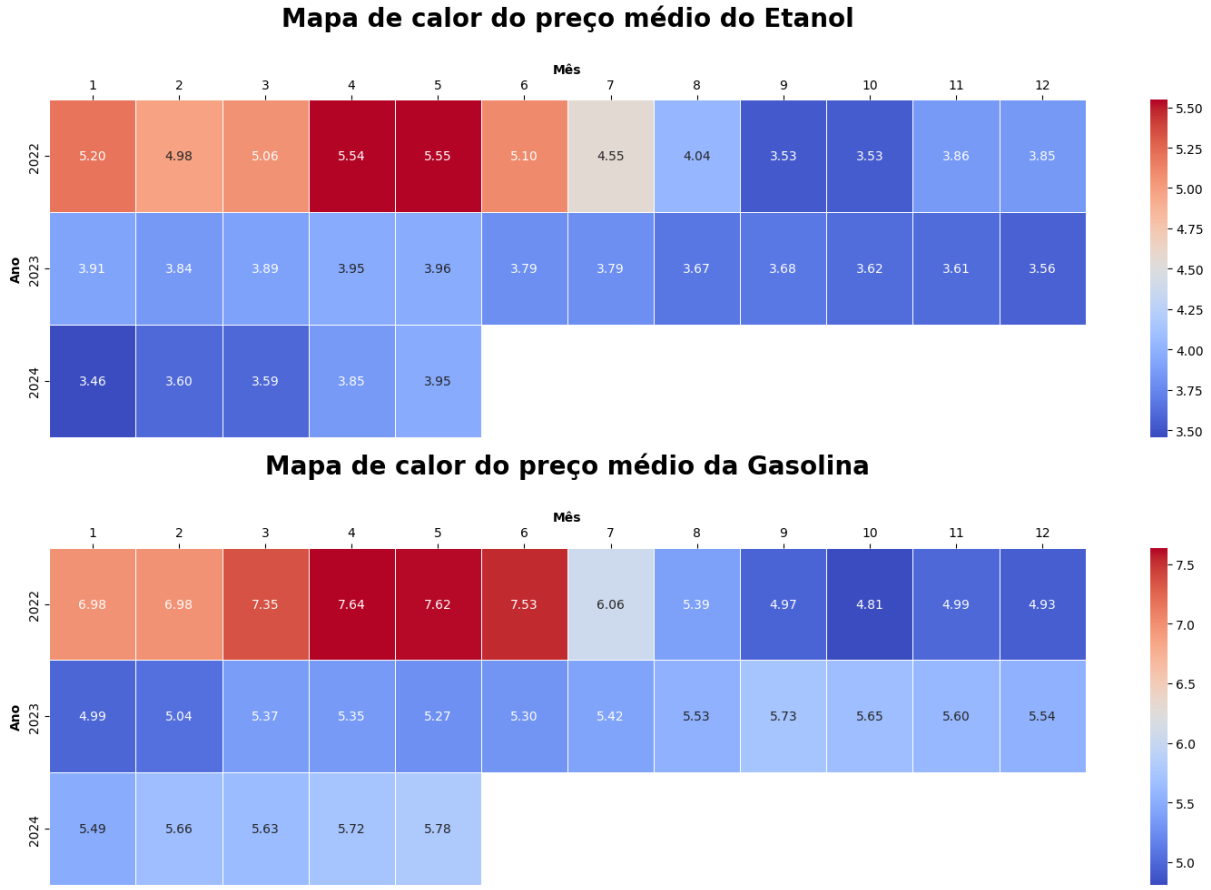
Os primeiros diagramas, cujo preço médio é limitado pelos valores máximos/mínimos do dia, é importante para avaliarmos fatores como dispersão e também os períodos de

maior volatilidade da variável. Por exemplo, no período de 2022, têm-se para o Etanol preços máximos na faixa de R$7,00, um comportamento extremo e que em dias próximos se repetiu. Além disso, no período completo da série para o Etanol é possível notar mais picos do que para a Gasolina, sugerindo que este combustível está sujeito a mais variações, podendo ser oriundos tanto de variáveis externas que o influenciam mais, características regionais da medida como a inclusão de postos em municípios que o custo de produção é maior ou simplesmente erros inerentes das medidas/pesquisa que por vezes pode não ter tido a possibilidade de coletar ocorrências representativas do preço do dia ao longo de todo o Estado.

Já o diagrama dos preços limitado pelos desvios padrões é importante para evidenciar a incerteza e a constante ocorrência das flutuações do valor da variável nos diferentes períodos analisados. Enquanto no período de maior volatilidade em 2022 têm-se a ocorrência dos valores extremos da variável com desvio padrão com certa variabilidade, nos períodos seguintes a tendência se estabiliza mas a variabilidade presente nos períodos anteriores se mantém.

Uma outra forma de visualizar os dois períodos descritos com os gráficos numéricos acima é através de um mapa de calor dos preços mês a mês, em cada ano do período analisado, conforme consta na figura abaixo.

Figura 21 – *Heatmap* do preço médio dos combustíveis nos meses do ano

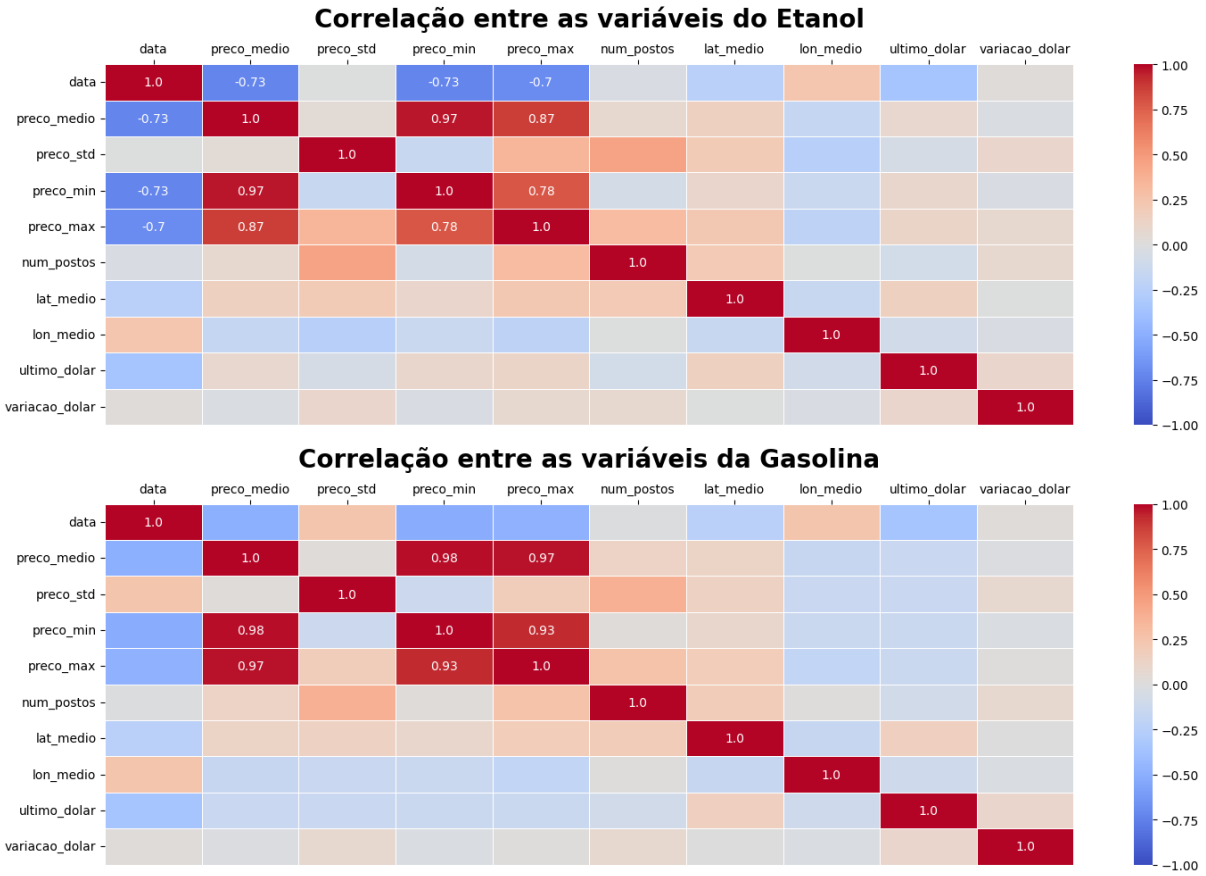


No gráfico é evidente a presença dos dois períodos distintos já citados no texto acerca das séries temporais acima, o período representado pelas cores mais “quentes” se estende até o meio do ano de 2022, enquanto nos períodos a partir dessa data têm-se que o valor da variável “esfria”, e se mantém com esse padrão mais estável durante o resto do período analisado. Essas variações são de notabilidade muito importante, visto que na etapa de modelagem podem ser fontes de perda de performance do modelo, que dependendo da sua construção pode apresentar dificuldades para generalizar os padrões de períodos “atípicos” e que não possuem grande ocorrência na série temporal.

Outra análise que é importante visando a etapa de modelagem é a aferição do valor das correlações estatísticas entre as variáveis não-categóricas que serão utilizadas para prever o valor do *target* (preço médio). Essa análise é não só importante para fornecer

possíveis candidatos que auxiliem na predição, mas também para entender se de fato faz sentido a associação entre duas variáveis no contexto do problema.

Figura 22 – Gráfico de correlações entre as variáveis não-categóricas da base de dados de combustíveis para cada combustível



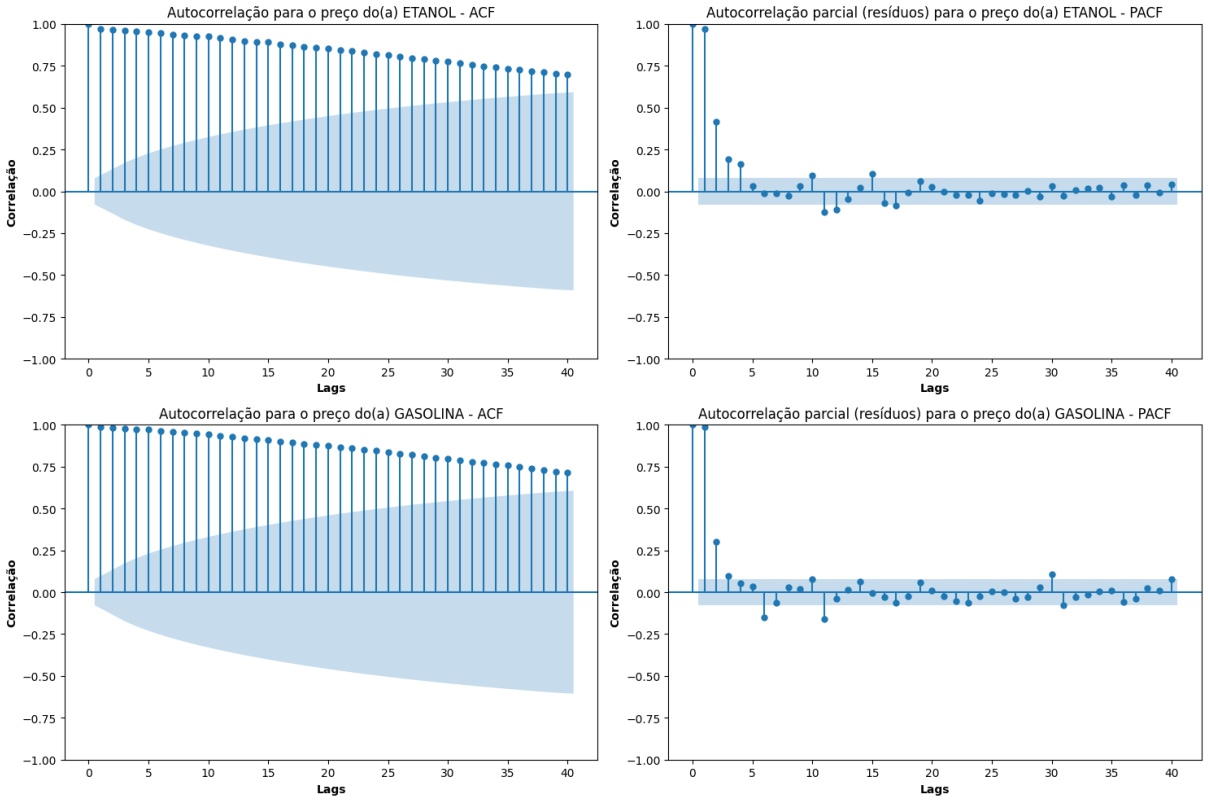
O cálculo das correlações foi realizado utilizando do método *corr()* do Python, nele são calculadas as correlações entre as variáveis de data ou numéricas da base de dados. Definiu-se como ponto de corte o valor de 0.7 para que a correlação tenha seu valor assinalado no mapa de calor acima, visto que a partir desse valor a correlação já é considerada significativa.

O diagrama revelou para ambos os combustíveis, além das correlações da variável com ela mesma, valores significativos de correlação entre as variáveis de preço (preco_medio, preco_min e preco_max), o que não fornece muita informação visto que são variáveis derivadas uma da outra. Além disso, têm-se que a data se correlaciona

negativamente com o preço médio, mais para o Etanol do que para a Gasolina, o que corrobora com a tendência de queda visualizada no gráfico de linha das séries temporais.

Por fim, para subsidiar mais recursos à próxima etapa, a de modelagem, têm-se a obtenção dos gráficos de autocorrelação da variável tempo, feita através da função de autocorrelação e autocorrelação parcial. Os gráficos de ACF (*autocorrelation function)* e PACF (*partial autocorrelation function)* são feitos com o auxílio da biblioteca *statsmodels*, com ampla gama de recursos estatísticos para subsidiar a análise. Esses gráficos consideram a correlação dos *lags* da variável tempo, começando de *lags* mais próximos até um valor pré-definido (que para este trabalho foi até o *lag* 40), no caso da PACF é contabilizado a influência dos *lags* após a eliminação do efeito dos demais *lags*. A obtenção dos diagramas foi feita com o auxílio das funções *plot_acf() e plot_pacf().*

Figura 23 – ACF e PACF *plots* para as séries temporais de preço médio dos combustíveis



O primeiro *lag* em todo gráfico sempre apresenta correlação máxima, visto que evidencia a correlação da variável com ela mesma no instante de tempo em que ela foi medida, sendo assim a análise se baseia a partir do *lag* de valor igual a 1, que representa a variável com ela um período antes. Nos gráficos ACF tanto para o Gasolina quanto o Etanol é possível notar um lento decaimento na correlação da variável conforme aumentam-se os *lags*, isso indica que, apesar das diversas possibilidades de influências externas no preço destes, há uma forte autocorrelação no preço médio em períodos longos, ou seja, os preços atuais influenciam de forma significativa os preços futuros. Essa longa memória no tempo da variável, ou seja, forte influência de períodos passados é um fator importante a se considerar no processo de modelagem, pois o peso que os *lags* possuem perante a predição da variável preço são relevantes.

De forma complementar, a análise dos *lags* sem efeito dos demais *lags* da série, realizada nos gráficos de PACF, revela uma forte correlação nos primeiros períodos decaindo de forma rápida em seguida, o que fornece uma indicação de que a dependência da série é capturada nos primeiros *lags*, sendo estes os mais importantes, tanto para a Gasolina quanto para o Etanol. Uma das importâncias dessas identificações e dos direcionamentos tomados a partir de quantos *lags* têm seu valor acima da linha de significância (em tom mais claro) é a definição dos parâmetros de modelos como ARIMA, ou que consideram somente médias móveis (MA) ou autorregressão (AR) [14].

Utilizando ainda da biblioteca *statsmodels* foi possível elaborar um último diagrama, a decomposição das séries temporais do Etanol e da Gasolina, obtendo suas componentes de Tendência e Sazonalidade, também importantes na etapa de modelagem. A obtenção dos diagramas foi possível através da função *seasonal_decompose().* Esta função pode realizar a decomposição da série a partir de dois modelos: **aditivo ou multiplicativo.** Ambos os modelos consideram que a série temporal é composta de 3 componentes: Tendência, Sazonalidade e Ruído. A forma como esses componentes interagem para compor a série é o que determina qual modelo mais se adequa ao uso na decomposição. Pode-se escrever então que os dois modelos para decomposição são dados por:

Modelo aditivo: **Y[t] = T[t] + S[t] + e[t]**

Modelo multiplicativo: **Y[t] = T[t] * S[t] * e[t]**

Sendo que:

- Y[t] representa a série temporal;
- T[t] representa a componente de **tendência;**
- S[t] é a componente de **sazonalidade;**
- e[t] é o **erro** ou **ruído** da série temporal

O modelo aditivo é comumente utilizado em séries temporais em que a sazonalidade possui padrões bem definidos, ou seja, não há muita variação no padrão de sazonalidade com o nível da série temporal. Além disso, quando a tendência não cresce ou decresce significativamente com o tempo, também se recomenda a utilização de modelos aditivos. Já para modelos multiplicativos, sua utilização é muito comum em séries temporais de economia, visto que nestas a sazonalidade e tendência são proporcionais ao nível da série temporal, e sofrem variação significativa com o tempo [13].

No problema abordado nesse trabalho será considerado o modelo **aditivo** para realização das decomposições das séries temporais do Etanol e da Gasolina, pois esses produtos apresentam sazonalidade padronizada.

Figura 24 – Decomposição da série temporal do Etanol

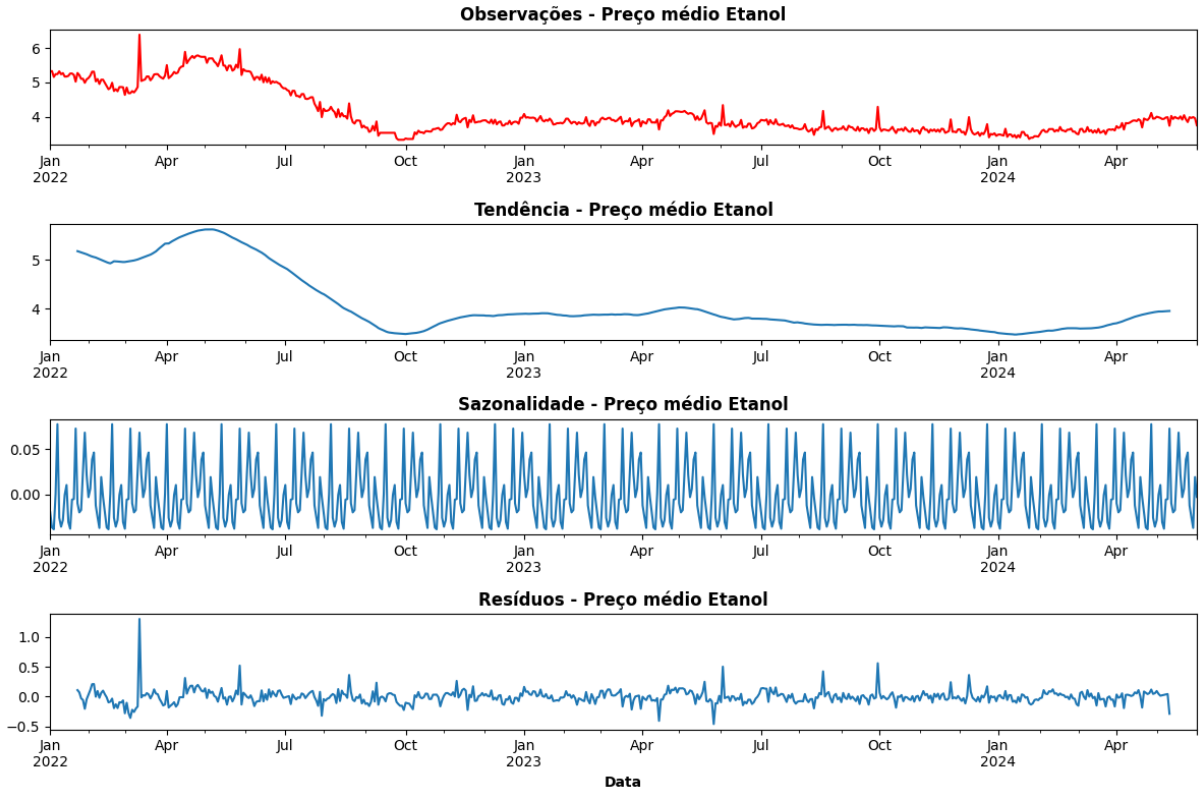
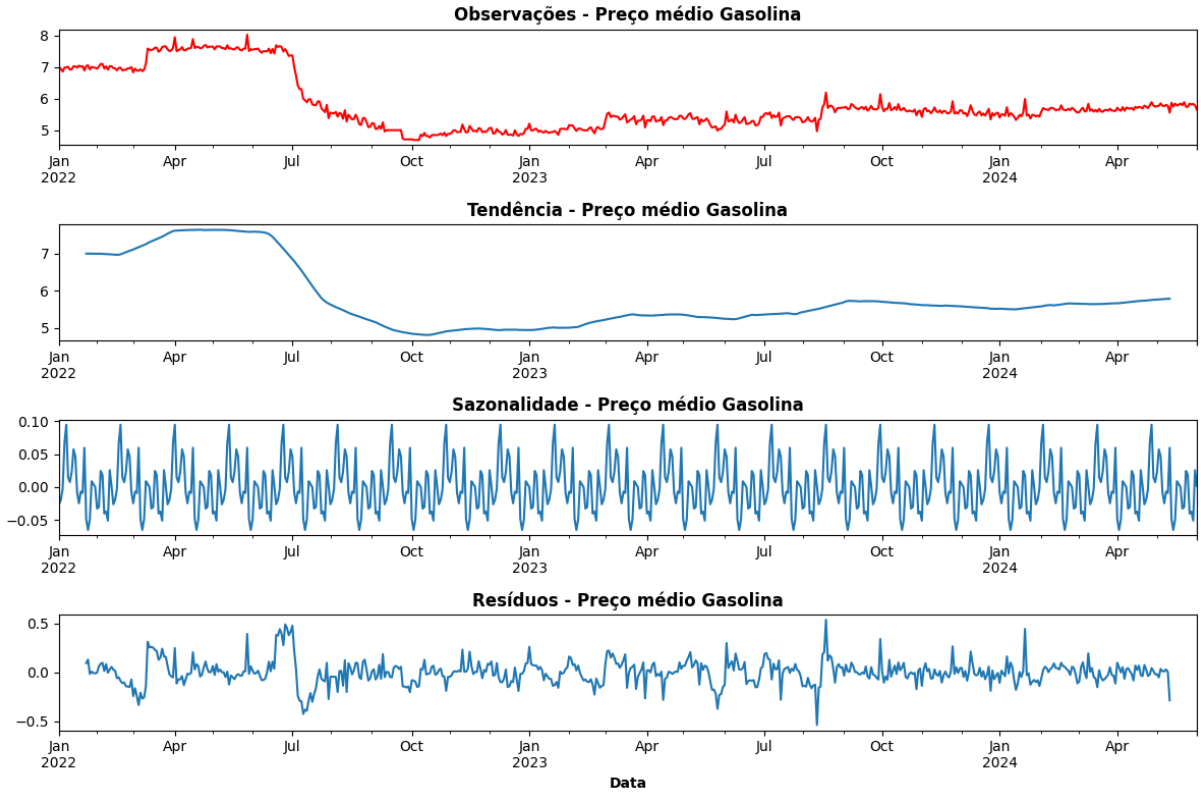


Figura 25 – Decomposição da série temporal da Gasolina



A decomposição das séries temporais dos combustíveis permite extrair padrões que haviam sido visualizados indiretamente nos gráficos anteriores, como a tendência e a sazonalidade das séries. Ao utilizar do método *seasonal_decompose()* os gráficos gerados são divididos em 4 *plots:*

- Observações: Este *plot* mostra a série temporal completa, no período analisado e incluindo todos os valores da variável alvo ao longo do tempo.
- Tendência: A componente que representa a direção que o valor da variável está seguindo com o tempo, seja crescente ou decrescente. A tendência é importante para revelar padrões a longo prazo da variável.
- Sazonalidade: É um padrão que se repete com certa frequência, seja ela diária, semanal, mensal etc. Está relacionada com as características da variável que está sendo representada, nesse caso, a sazonalidade do preço dos combustíveis é aproximadamente semanal/mensal, visto que variáveis externas como custos de produção, oferta e demanda afetam a variável nessa frequência.
- Resíduos: Referem-se as flutuações aleatórias que restam após a remoção da tendência e sazonalidade, também chamados de ruído, quanto maior a presença destes, mais se torna difícil identificar os padrões da série.

Falando acerca do Etanol, é possível notar a queda significativa no primeiro semestre de 2022, através da componente de tendência. Posteriormente, a tendência se estabiliza e sofre poucas flutuações, fruto do período pós-pandêmico (2023 e 2024) onde os preços tiveram menos volatilidade. Apesar da pouca variação na tendência da série em períodos mais recentes, é possível notar um padrão na sazonalidade do Etanol, com uma alta frequência de ocorrência (ciclos semanais ou mensais), fator que gera uma flutuação consistente do preço do combustível e que gera uma componente significativa para modelos de previsão. A última componente da decomposição, os resíduos, contém a parte que não fora explicada pela tendência e sazonalidade anteriormente, e para o Etanol têm um comportamento aproximadamente estacionário, indicando que as duas componentes anteriores já capturam grande parte do comportamento do combustível.

A Gasolina apresenta padrões em suas componentes semelhantes ao do Etanol, reforçando que certas características afetam os combustíveis no geral, e não separadamente. Na componente de tendência, é perceptível uma queda mais acentuada no preço médio no período citado para o Etanol, algo em torno de 30%, mas seguido de uma estabilização que também ocorreu com o outro combustível analisado. Na sazonalidade têm-se um padrão bem definido, trazendo mais uma vez uma frequência de ocorrência, fator que permite padronizar os mesmos modelos e utilização dessas componentes para ambos os combustíveis. Por fim, os resíduos da série temporal da Gasolina são mais voláteis, sugerindo mais ruído na série e componentes que não são completamente obtidas pela tendência e sazonalidade, tal fator pode ser atribuído às já citadas variáveis externas referentes a esse produto.

Com o fim da Análise Exploratória dos Dados é possível adentrar à etapa de elaboração de modelos de Machine Learning, neste trabalho busca-se obter um horizonte de previsão significativo para os preços médios tanto da Gasolina quanto do Etanol. Apesar de terem características semelhantes, os combustíveis serão tratados em modelos distintos, onde cada treino será realizado com seu devido *dataset* e os resultados desta etapa de análise exploratória servirão como base para construção de variáveis e modelos escolhidos.

5. Criação de Modelos de Machine Learning

O processo de desenvolvimento de modelos de aprendizado de máquina (*Machine Learning*), sobretudo voltado para problemas de séries temporais, é um problema complexo e que abrange diversas etapas. A dependência temporal dos dados exige um cuidado e considerações específicas no processo de modelagem e suas etapas. Cada uma dessas etapas desempenha um papel crucial na eficácia do modelo e na sua capacidade de generalizar para dados que nunca foram vistos (dados reais ou que simulem a realidade). Neste contexto, o objetivo é treinar um algoritmo que reconheça os padrões, especialmente os pertinentes à variável temporal, em um conjunto de dados dito conjunto de treino e a seguir fazer com que ele possa fazer previsões, chamadas de *forecast*, dos dados no futuro.

A seguir, detalhamos as principais etapas envolvidas no processo de criação de um modelo de *Machine Learning*:

A definição do problema – *Machine Learning* pode ser aplicado a inúmeros casos de uso, ter um escopo bem definido e o problema que se deseja resolver é essencial para uma boa modelagem. Tais definições pautam decisões como quais modelos utilizar ou no caso deste trabalho, responder qual será o horizonte de previsão, quais variáveis serão explicativas e quais desejamos prever.

Escolha do(s) modelo(s) de aprendizado de máquina – Existem várias categorias de algoritmos de *Machine Learning* disponíveis para escolha. A decisão de qual utilizar parte especialmente da resposta de qual problema desejamos resolver ou pergunta responder. É com esse subsídio que é possível definir se os algoritmos serão voltados para uma classificação, uma regressão ou também uma segmentação dos dados. Para cada tipo de problema é possível escolher **algoritmos supervisionados, não-supervisionados, de aprendizado por reforço ou até redes neurais.**

Pré-processamento de dados – Etapa que pode preceder a anterior, mas também é afetada pela escolha do modelo, visto que é nessa etapa que escolhemos as transformações que faremos nos dados para que se adequem aos modelos, como por exemplo o *encoding* de variáveis categóricas. Além das transformações, também é feita a divisão da base de dados em treino e teste (caso seja um problema supervisionado), são criadas variáveis

explicativas derivadas de *insights* obtidos na análise exploratória de dados, especialmente para problemas de séries temporais onde são importantes a criação de *lags,* médias móveis e suavizações das componentes. Por fim também é nessa etapa onde é realizada o tratamento de *outliers* e de dados faltantes.

Treinamento e avaliação do modelo – São fornecidos então ao modelo os dados históricos, denominados dados de **treino**, para que o modelo possa aprender os padrões destes buscando minimizar uma função de perda e ajustar seus parâmetros de acordo com a o valor da saída, visto que na base de treino são fornecidas as saídas esperadas para a variável *target.* É no treinamento que os padrões de sazonalidade, tendência e características temporais são "aprendidos" pelo modelo. Após o treinamento do modelo é feita a avaliação da performance dele, utilizando da base de **teste.** Essa base busca emular os dados reais, ou seja, para o modelo não há os valores da variável a ser prevista, de modo que o mesmo deve buscar generalizar para dados novos e ser capaz de identificar os padrões de forma semelhante aos de treino. Utiliza-se uma métrica adequada para avaliação da performance do modelo juntamente com técnicas de validação cruzada, para evitar o *overfitting.*

Neste trabalho, por conta da característica temporal do problema, não é possível utilizar um método de *cross-validation* simples com divisão aleatória dos dados no momento do treino. Ao invés disso é necessário **respeitar o ordenamento temporal**, empregando técnicas que considerem esse fator no momento do treinamento, para o nosso caso utilizamos a validação de séries temporais (TSCV).

5.1. Escolha de modelos para séries temporais

Dentre as diversas abordagens existentes para a escolha de modelos que se adequem ao problema de séries temporais neste trabalho foram escolhidos especialmente modelos de **árvore de decisão** e **autorregressivos.** Mais especificamente, o modelo baseado em árvore de decisão: **Random Forest**, o modelo baseado em *gradiente boosting*: **XGBoost** e o modelo baseado em variáveis autorregressivas e exógenas: **SARIMAX.**

Os modelos *Random Forest e XGBoost* foram escolhidos para o problema deste trabalho pois funcionam de forma robusta à *outliers*, fator presente nas variáveis que

desejamos prever para os combustíveis, e permitem que os dados não precisem sofrer transformações de **estacionariedade** e reduzir as relações para um grau linear, além disso são modelos que lidam bem com padrões complexos que envolvem muitas variáveis explicativas. Por fim, são modelos que tratam bem o *overfitting*, fornecendo subsídio para evitá-lo.

O modelo SARIMAX, uma variação do ARIMA é uma escolha tradicional para problemas de séries temporais, visto que permite a captura de padrões de tendência e sazonalidade dos dados históricos. A variação SARIMAX permite a inclusão das componentes de **sazonalidade** no treinamento, além de variáveis exógenas (externas), de suma importância para o problema que tratamos, no qual somente a data não seria suficiente para fornecer todos os padrões pertinentes ao problema de negócio. Por fim, modelos baseados em ARIMA possuem fácil interpretação.

Mesmo com as escolhas acima, problemas de séries temporais permitem a utilização de diversas famílias de algoritmos, como por exemplo:

Redes Neurais Recorrentes (RNN e LSTM): Classes de redes neurais com arquiteturas projetadas para lidar com dados sequenciais. Se diferenciam das arquiteturas clássicas de redes neurais como as *feed-forward* ou convolucionais por terem em sua arquitetura mecanismos para que a rede possa “lembrar” de informações vistas anteriormente, isso essencialmente é feito utilizando de estados internos que são mantidos como termos de longa memória e são recuperados quando necessário.

Regressões: Modelos clássicos que buscam obter relações lineares ou polinomiais entre os dados e que podem ser adaptados para respeitar o ordenamento temporal dos dados. O modelo de regressão busca ajustar uma função que minimize o erro médio das previsões, de acordo com uma função de perda.

5.1.1. Random Forest

Random Forest (ou floresta aleatória em tradução livre) é um modelo baseado em múltiplas árvores de decisão (*Decision Tree)*. As árvores de decisão são modelos que podem ser aplicados às tarefas de classificação ou regressão, e funcionam com base em nós, onde cada um representa uma decisão construída a partir de um ou mais atributos da base de

dados, após os nós existem as folhas da árvore, que são representações das escolhas dos valores feitos para a variável ou classe *target*. O *Random Forest* foi introduzido como uma forma para melhorar a precisão e mitigar o *overfitting* de árvores de decisão, e o faz utilizando do conceito de agregação de múltiplas árvores de decisão independentes. Ao treinar várias árvores com subconjuntos diferentes de dados e características, o *Random Forest* combina as previsões de todas as árvores para produzir uma previsão final por meio de uma média.

No modelo é utilizado o método de *Bootstrap Aggregation – Bagging* durante o treinamento. Neste método, para cada árvore do modelo, é gerado um conjunto de dados diferente por meio de amostragem aleatória com reposição. Além disso, ao construir as árvores, o algoritmo escolhe um subconjunto aleatório de características em cada nó, o que gera diversidade entre as árvores e reduz a correlação entre elas, evitando assim o *overfitting* em árvores específicas. Além disso, existem hiperparâmetros específicos do modelo que podem ser ajustados para melhorar a performance do modelo, como por exemplo o **número de árvores, a profundidade máxima de cada árvore, o número mínimo de amostras em cada nó e o número máximo de atributos utilizado para que seja criado um nó.** Todos esses hiperparâmetros podem ser ajustados após a avaliação do modelo, a fim de melhorar sua precisão e generalização, visto que esses ajustes permitem com que se controle justamente como é feito o treinamento do modelo, a divisão dos nós e a tomada de decisão com base nos atributos.

Por fim, no treinamento o modelo utiliza de uma função de custo para ajustar os seus parâmetros no treinamento, essa função pode ser o erro quadrático ou absoluto médio (MSE e MAE respectivamente) ou Entropia em cada nó, dependendo se a tarefa é classificação ou regressão.

Vantagens: Facilmente interpretável, os modelos baseados em árvore fornecem uma estrutura fácil de ser entendida para explicar como a previsão foi feita, pois é necessário apenas entender os nós seguidos e a folha final. Além disso, o *Random Forest* é robusto à *overfitting* visto que utiliza técnicas de *ensemble* de várias árvores independentes. Por fim também é uma vantagem desse tipo de modelo não ser exigido normalização dos dados e o

tratamento de variáveis categóricas, visto que modelos de árvore lidam bem com variáveis mistas.

Desvantagens: O ajuste dos hiperparâmetros, especialmente para valores muito altos podem onerar a execução do modelo, exigindo muito computacionalmente e tornando o modelo mais complexo conforme novos nós são formados.

5.1.2. XGBoost

Extreme Gradient Boosting, ou *XGBoost* é uma implementação eficiente da técnica de B*oosting* em modelos que utilizam de árvores de decisão para classificação ou regressão. O *Gradient Boosting* é uma técnica de aprendizagem em que os modelos (árvores de decisão) são treinados sequencialmente, e cada modelo subsequente tenta corrigir os erros cometidos pelo modelo anterior, cada modelo é dito um modelo fraco, mas que com seus resultados atuando de forma sequencial é possível atingir resultados significativos.

O método utilizado para ajuste do modelo no seu treinamento segue o **gradiente do erro**, o que significa que o modelo busca minimizar o quão longe as previsões estão do valor real, adaptando as árvores para minimizar essa métrica. O uso do gradiente é uma técnica fortemente empregada para melhoria da performance de modelos pois é extremamente prático em encontrar mínimos locais e até globais para funções de perda complexas, o que o torna uma técnica essencial para casos em que modelos simples não são suficientes. Além da utilização do gradiente, no treinamento o *XGBoost* emprega técnicas de regularização afim de evitar o *overfitting.*

Geralmente as funções de perda utilizadas no processo de treinamento são o erro quadrático médio (MSE) ou o *log-loss.* Os hiperparâmetros presentes no modelo são semelhantes ao do *Random Forest* visto que aqui também são utilizadas árvores de decisão, porém com otimização atrelada ao **gradiente do erro**, dentre esses parâmetros têm-se o **número de árvores, a profundidade das mesmas, o learning rate** que controla o tamanho dos passos dados a cada iteração, o que pode acelerar ou impedir a convergência para a minimização do erro.

Vantagens: O *XGBoost* é otimizado computacionalmente para utilização em dados complexos, sejam com muitas instâncias ou com muitos atributos. Por utilizar da técnica de *Gradient Boosting* por vezes apresenta uma diversificação e flexibilidade maior que o R*andom Forest.* Por fim, o modelo traz consigo técnicas robustas para evitar o *overfitting* como regularização por exemplo.

Desvantagens: Com maior complexidade e maior número de parâmetros o modelo pode se tornar mais difícil de ser ajustado e implementado. Há uma perda na interpretabilidade conforme o número de estimadores e nós aumenta, visto que o modelo se torna mais complexo afim de capturar padrões complexos, mas perde clareza ao utilizar de muitas regras para fazê-lo.

5.1.3. SARIMAX

Autoregressive integrated moving average (ARIMA) ou simplesmente modelo autorregressivo integrado de médias móveis é um método ou modelo estatístico indicado para análise de dados que apresentam tendências e sazonalidades, de modo que possam ser feitas transformações para tornar esses dados **estacionários,** isto é, com valores como variância e médias constantes diante da variável temporal. Os modelos ARIMA generalizam modelos ARMA, que consideram duas componentes no modelo:

A componente **autorregressiva (AR)** que refere-se à capacidade da variável de tempo ter relação consigo mesma nos períodos atuais e períodos anteriores, para isso busca-se com análises de autocorrelação entender quais períodos passados são mais relevantes para a predição atual.

A outra componente são as **médias móveis (MA)** que consistem da utilização da contribuição dos erros (resíduos) de períodos anteriores para ajuste nas previsões futuras. As componentes autorregressivas e médias moveis são então **integradas (I)** através da diferenciação dos valores da variável temporal para tornar a série estacionária.

O modelo SARIMAX consiste na utilização dos métodos ARIMA mas com uma extensão que inclui a contribuição de padrões da sazonalidade da série temporal (*S – Seasonality)* e também as variáveis ditas exógenas (*X – Exogenous Variables)* que podem

influenciar na previsão da variável *target* dependendo do tipo de problema. Neste trabalho, em especial na análise exploratória de dados, foi possível notar que variáveis externas apresentam forte influência no preço dos combustíveis, visto que tal mercado é sensível à fatores externos que não só a variável temporal.

Vantagens: Por ser um modelo clássico para séries temporais, possui ótima performance se capturar de forma eficiente os padrões sazonais e de tendência. Além disso fornece fácil interpretação ao deixar explícito as relações temporais.

Desvantagens: Pode apresentar limitações para séries temporais com padrões mais complexos e que fujam muito da linearidade. Além disso, séries que trazem maior complexidade através de longos períodos ou muitos dados podem comprometer a performance do modelo.

5.2. Pré-processamento e *Feature Engineering*

A preparação dos dados para a etapa de treinamento dos modelos é uma tarefa de suma importância, ela é necessária pois muitos modelos por vezes são construídos para funcionarem apenas com variáveis numéricas como *input*, ou também variáveis que são normalizadas dentro de certo intervalo de dados. Transformações, *encodings,* criação de variáveis *dummy* e divisão entre base de treino e teste são exemplos de etapas do pré-processamento e *feature engineering*.

No contexto do problema abordado neste trabalho, de séries temporais, esta etapa se destaca pela criação de novas variáveis derivadas da variável temporal como *lags* e médias móveis por exemplo, que são importantes para a captura das componentes das séries como tendência e sazonalidade, além de identificar padrões de curto prazo, algo que se mostrou relevante pela análise exploratória dos nossos dados e suavizar os ruídos das séries. A criação dessas novas variáveis foi feita com o auxílio da biblioteca *open-source* em Python *MLForecast* de propriedade da empresa Nixtla [14], que traz consigo diversas funções para obter tais transformações temporais nos dados, bastando adaptá-los para as funções.

Primeiramente foi necessário adaptar variáveis da base de dados que a biblioteca exige que estejam em uma nomenclatura específica, foi o caso da variável temporal, da

variável *target* e da variável que identifica a série temporal que respectivamente se tornaram ‘ds’ (era ‘data’ previamente), ‘y’ (previamente ‘preco_medio’) e ‘unique_id’ (‘tipo_comb’). Após a adequação, criou-se um objeto do *MLForecast* que permitiu a criação das variáveis temporais como *lags,* médias móveis etc. As variáveis bem como sua criação estão expressas nas figuras abaixo:

Figura 26 – Criação de *features* utilizando a biblioteca *MLForecast*

```
## RENOMEANDO AS COLUNAS DE DATA, VARIÁVEL TARGET E IDENTIFICADOR DA SÉRIE TEMPORAL PARA SE ADEQUAR AO MLFORECAST
df_fuels = df_fuels.rename(columns={'data': 'ds', 'preco_medio': 'y', 'tipo_comb': 'unique_id'})
```

0.0s

```
## DEFININDO OS OBJETOS PARA CÁLCULO DAS FEATURES COMO DIFERENÇA, LAGS E MÉDIA MÓVEL

### FUNÇÃO DE DIFERENCIAÇÃO DEFINIDA UTILIZANDO O NUMBA (ADEQUADO AO MLFORECAST)
@njit
def difference(x, lag):
    diff_x = np.full_like(x, np.nan)
    for i in range(lag, len(x)):
        diff_x[i] = x[i] - x[i-lag]
    return diff_x

### OBJETO DO MLFORECAST PARA CÁLCULO DE FEATURES COMO LAGS, MÉDIA MÓVEL E VARIÁVEIS DE DATA
get_features = MLForecast(
    models=[],
    freq='D',
    lags=[1,7],
    lag_transforms={
        1: [RollingMean(window_size=3), RollingMean(window_size=7), (difference, 1), (difference, 7)]
    },
    date_features=['dayofweek', 'month', 'year', 'day'],
    num_threads=2
)

### APLICAÇÃO DO OBJETO NO DATAFRAME PARA OBTENÇÃO DAS FEATURES
df_fuels = get_features.preprocess(df_fuels, id_col='unique_id', time_col='ds', target_col='y', static_features=[])
```

Variáveis finais do *dataset* de combustíveis para treinamento dos modelos

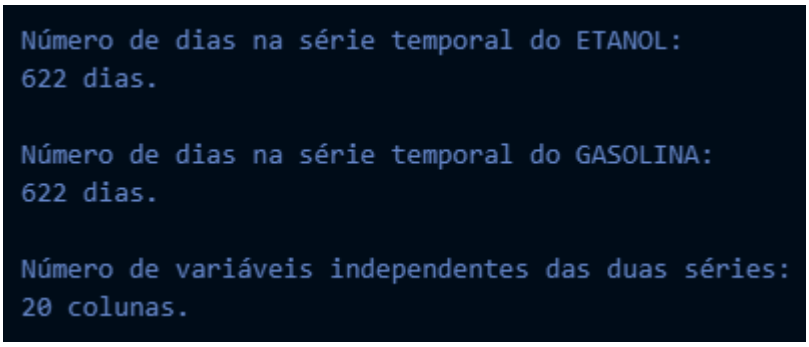
Nome da coluna/campo	Tipo
ds	*Datetime*
unique_id	*String*
y	*Float*
preco_std	*Float*
preco_min	*Float*
preco_max	*Float*

num_postos	*Int*
lat_medio	*Float*
lon_medio	*Float*
ultimo_dolar	*Float*
variacao_dolar	*Float*
lag1	*Float*
lag7	*Float*
rolling_mean_lag1_window_size3	*Float*
rolling_mean_lag1_window_size7	*Float*
difference_lag1_lag1	*Float*
difference_lag1_lag7	*Float*
dayofweek	*Int*
month	*Int*
year	*Int*
day	*Int*

Como é possível notar, foram criadas 10 novas variáveis explicativas derivadas da variável temporal. Tal abordagem é de suma importância para fornecer ao modelo os padrões sazonais e de tendência da série. Dentre as escolhas de possíveis derivação da variável temporal foram escolhidas: os *lags* de 1 e 7 períodos (dias), a diferença de 1 e 7 períodos (dias), a média móvel em uma janela de 3 e 7 períodos e a decomposição da data em dia da semana, mês, ano e dia.

Para os dados de combustíveis que utilizaremos no treinamento dos modelos não foi necessária mais nenhuma transformação, visto que a única variável categórica (que geralmente necessita ser transformada) é o identificador de se o combustível é Etanol ou Gasolina, o que efetivamente não contabilizará como variável explicativa para os modelos, as demais são numéricas, já sendo suficiente para modelos como ARIMA ou *Random Forest e XGBoost*. Dessa forma foi possível então, separar cada série temporal individualmente (Etanol e Gasolina) para realizar a divisão da base em treino e teste.

Figura 27 – Quantidade de dias e variáveis na base de dados do Etanol e Gasolina



5.3. Divisão em treino e teste e *Cross-validation*

A separação da base de dados em base de treinamento e base de testes (ou avaliação) foi a etapa final de processamento dos dados de combustíveis para a modelagem neste trabalho. Esse *split* é importante em qualquer projeto de Machine Learning que empregue modelos **supervisionados**, pois dessa forma é garantido que o processo de treinamento do modelo é realizado com uma parte dos dados que tragam os padrões que se deseja que o modelo aprenda e que também outra parte dos dados sejam utilizados para "testar" a performance real do modelo, ou seja, simular como seria sua performance em dados nunca vistos (dados reais).

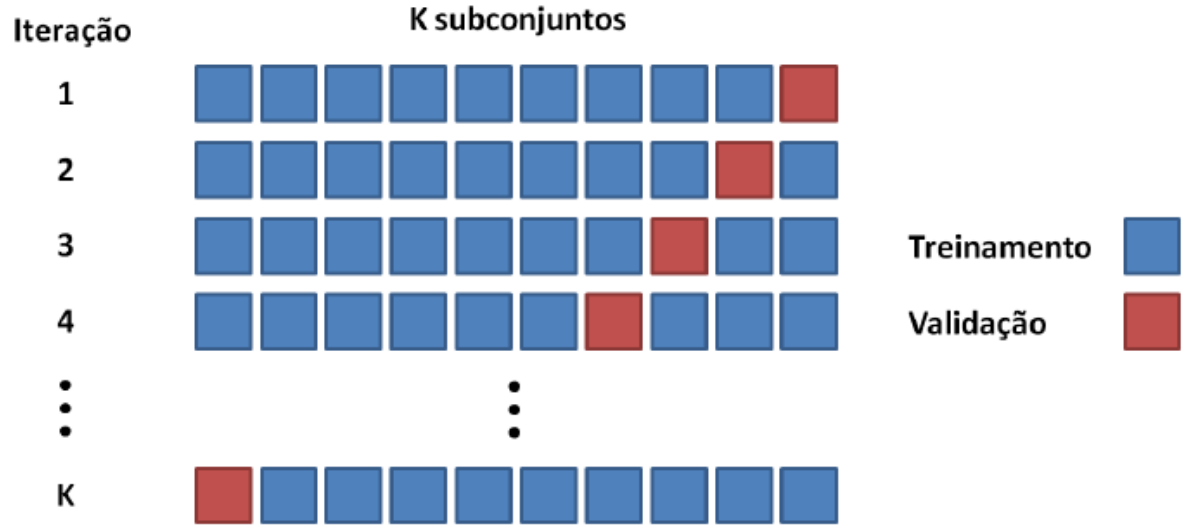
Dividir os dados em treino e teste não deve ser feito de maneira negligente, afinal é importante que se tenham amostras representativas em cada parte dos dados (treino e teste). A fim de garantir isso são empregadas técnicas de separação dos dados de forma estratificada, o que garante em muitos casos de classificação e regressão que teremos amostras significativas em ambas as bases. No problema deste trabalho, devemos garantir que na nossa separação **a ordem temporal seja preservada**, isto é, não deve haver vazamento de dados futuros para bases que tratam do passado, visto que se deseja que o modelo aprenda padrões históricos dos dados para prever valores futuros.

O modo utilizado para dividir os dados de combustíveis do Etanol e Gasolina para preservar a ordem temporal foi o objeto *TimeSeriesSplit* da biblioteca *scikit-learn*, esse objeto permite não só a divisão entre treino e teste respeitando a ordem temporal, mas também gera *folds* dos dados que também respeitam essa ordem. Esses *folds* serão úteis para que seja realizado o *cross-validation* dos dados no treinamento do modelo, outro

processo de suma importância em um projeto de Machine Learning para evitar a ocorrência de *overfitting*.

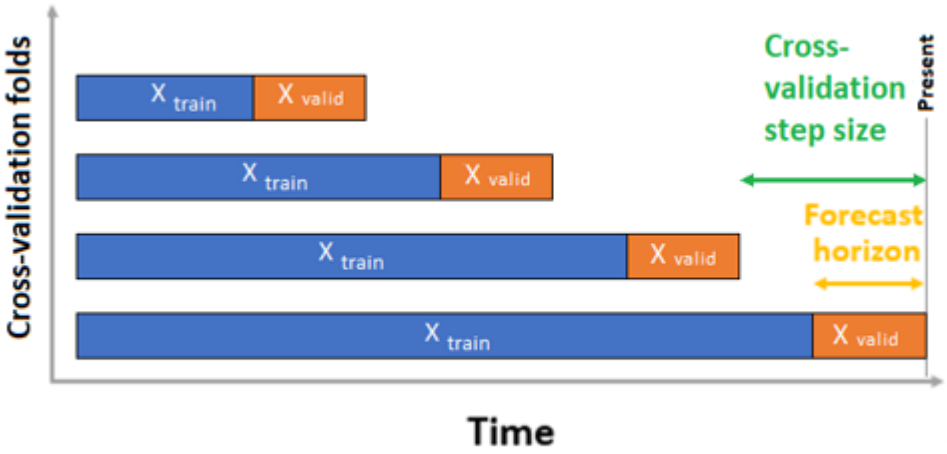
Cross-validation ou validação cruzada é uma técnica utilizada para avaliar a performance de modelos de *Machine Learning* de forma robusta e confiável. Para séries temporais, no entanto, a forma mais comum de *cross-validation* não pode ser aplicada, uma vez que as observações ao longo do tempo estão correlacionadas e devem ter sua ordem temporal preservada. Isso exige uma abordagem diferente conhecida como **validação cruzada temporal**, que respeita a ordem cronológica dos dados.

Figura 28 – Validação cruzada clássica dos dados sem ordem temporal



Fonte: Eric Couto – ericcouto.wordpress.com

Figura 29 – Validação cruzada para séries temporais



Como é possível notar nas figuras acima, o processo de separação da base em treino e teste e a criação dos *folds* para validação cruzada compõem um processo só, que é feito de forma iterativa e incremental para o caso de séries temporais, método esse chamado de **validação temporal por janela expansiva.** Apesar da utilização neste trabalho, ele não é o único método de *cross-validation* para séries temporais, existem outros como uma simples separação num ponto fixo respeitando a ordem temporal ou validação com janela deslizante, que podem ser mais aplicáveis a outros problemas envolvendo dados com data.

Figura 30 – *TimeSeriesSplit* para divisão da base em treino e teste

```
### DIVISÃO EM TREINO E TESTE DOS DATAFRAMES DE COMBUSTÍVEIS SEGUINDO A ESTRATÉGIA DE VALIDAÇÃO CRUZADA DE SÉRIES TEMPORAIS
tscv = TimeSeriesSplit(n_splits=4, test_size=90)
folds_treino_gasolina, folds_treino_etanol = [], []
folds_teste_gasolina, folds_teste_etanol = [], []

for train_index, test_index in tscv.split(X_gasolina):
    folds_treino_gasolina.append(X_gasolina.iloc[train_index])
    folds_teste_gasolina.append(X_gasolina.iloc[test_index])

for train_index, test_index in tscv.split(X_etanol):
    folds_treino_etanol.append(X_etanol.iloc[train_index])
    folds_teste_etanol.append(X_etanol.iloc[test_index])
```

O processo realizado pelo objeto *TimeSeriesSplit* pode ser descrito nas seguintes etapas:

1. **Divisão dos dados em conjuntos Treino e Teste**: Os dados são divididos em várias partições/*folds*, respeitando a ordem temporal dos dados. O primeiro subconjunto de treino contém os dados mais antigos, e o teste contém dados que vieram logo após esse período. Vale lembrar que é nesta etapa que são definidos quantos *folds* serão gerados e o tamanho do conjunto de teste, que será **fixo**.
2. **Treinamento incremental**: Na etapa de treinamento do modelo cada fold treino-teste é passado ao mesmo, a fim de treinar e avaliar as previsões naquele período. No segundo *fold*, o modelo é treinado com uma quantidade maior de dados, ou seja, o conjunto de treino agora inclui mais observações temporais (acrescenta os dados de teste do primeiro *fold* ao treino). O conjunto de teste continua sendo o próximo bloco cronológico de dados.
3. **Repetição do processo**: Para todos os *folds* da divisão é repetido o processo, sempre mantendo a ordem cronológica dos dados e avançando no tempo para cada iteração.
4. **Avaliação**: Após o treino-teste do modelo para todos os *folds*, se calcula a métrica de erro escolhida para cada *fold*, e o desempenho do modelo é avaliado com base na média dessas métricas. Isso permite com que possam ser avaliados processos como *overfitting* e permitir ver como o modelo performa em diferentes períodos dos dados.

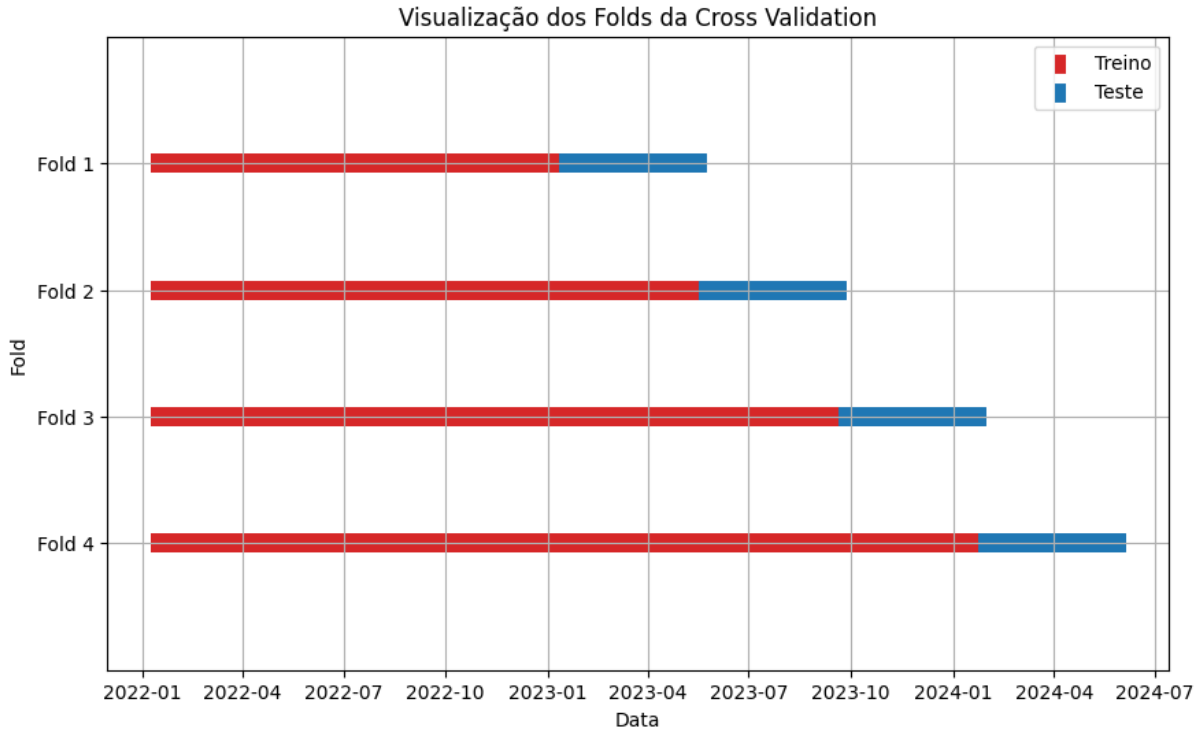
Na figura abaixo é possível visualizar as datas que compõe os *folds* para os dados de combustíveis. Neste trabalho foi decidido que a divisão da base seria feita em 4 subconjuntos, onde a base de teste teria sempre 90 dias.

Figura 31 – Datas de início e fim dos *folds* da validação cruzada para as bases de treino/teste

```
TREINO (GASOLINA e ETANOL)
-----------------
FOLD_0: 262 datas       Inicio: 2022-01-13        Fim: 2023-01-13
FOLD_1: 352 datas       Inicio: 2022-01-13        Fim: 2023-05-19
FOLD_2: 442 datas       Inicio: 2022-01-13        Fim: 2023-09-22
FOLD_3: 532 datas       Inicio: 2022-01-13        Fim: 2024-01-26

TESTE (GASOLINA e ETANOL)
-----------------
FOLD_0: 90 datas        Inicio: 2023-01-16        Fim: 2023-05-19
FOLD_1: 90 datas        Inicio: 2023-05-22        Fim: 2023-09-22
FOLD_2: 90 datas        Inicio: 2023-09-25        Fim: 2024-01-26
FOLD_3: 90 datas        Inicio: 2024-01-29        Fim: 2024-05-31
```

Figura 32 – Visualização gráfica das divisões da base de treino e teste



5.4. Treinamento dos modelos

Os 3 modelos escolhidos para serem utilizados para buscar prever o preço médio dos combustíveis Etanol e Gasolina foram o SARIMAX (SARIMA + variáveis exógenas), *Random Forest* e *XGBoost*. Todos são modelos que podem ser aplicados para problemas de

regressão, ou seja, um problema que busca prever um valor contínuo, seja dependente temporalmente ou não.

Com a criação das novas variáveis e dos índices do *cross-validation* na etapa passada, foi possível realizar o processo de treinamento e avaliação de cada um desses modelos para os dados de combustíveis. A abordagem utilizada consistiu em separar as séries do Etanol e da Gasolina em dois *dataframes*, e para cada um realizar em cada modelo o treinamento com a base de **treino** e a avaliação (previsão) com a base de **teste.** Dessa forma, cada um dos 3 modelos foi treinado de forma diferente duas vezes, uma para cada combustível. Essa abordagem é importante pois mesmo que os combustíveis apresentem características muito parecidas nos dados, é possível que algum modelo performe melhor nas previsões de um combustível do que outro, apesar de não ser o esperado.

A última etapa de treinamento dos modelos após a primeira avaliação consiste na otimização de hiperparâmetros, essa é uma etapa final que fornece uma espécie de "ajuste fino" à performance dos melhores modelos seguindo uma métrica escolhida. O algoritmo que utilizaremos para realizar essa busca pelos melhores parâmetros de cada modelo é o *optuna*, um *framework* em *Python* voltado para esta tarefa. Com a busca do *optuna* será possível obter os melhores parâmetros que minimizam nossa métrica de erro.

As métricas escolhidas para avaliar e interpretar a qualidade da previsão dos nossos modelos candidatos foram a **Raiz do Erro Quadrático Médio** (*Root Mean Squared Error – RMSE*) como métrica primária, que desejamos otimizar e minimizar e o **Erro Absoluto Médio** (*Mean Absolute Error – MAE*) como métrica secundária, para auxiliar na avaliação da performance dos modelos).

Figura 33 – Cálculo da Raiz do Erro Quadrático Médio (RMSE)

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

where:

- y_i is the actual value
- $\hat{y}_i$ is the predicted value
- n is the number of observations

Figura 34 – Cálculo da Erro Absoluto Médio (MAE)

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

where:

- y_i is the actual value
- $\hat{y}_i$ is the predicted value
- n is the number of observations

Ambas as métricas são comumente utilizadas para problemas de regressão. A Raiz do Erro Quadrático Médio foi escolhida devido a algumas características, como por exemplo sua sensibilidade à *outliers*, isso significa que previsões que fujam muito do padrão dos valores reais serão mais penalizadas. Isso vai de encontro com o objetivo da elaboração desses modelos, afinal a previsão do preço de combustíveis não pode diferir muito da realidade considerando que este é um produto com alta sensibilidade na mudança de seu preço, poucos centavos no preço médio do litro podem gerar impactos significativos. Outra característica importante do RMSE é que ele está nas mesmas unidades da variável *target*, o preço médio do combustível, dessa forma é possível visualizar de forma clara qual é o erro

médio do modelo já no número do RMSE. Por fim, o Erro Absoluto Médio é utilizado aqui como métrica auxiliar para entender como estão de forma absoluta os erros do modelo, visto que essa métrica traz uma média simples da diferença entre o previsto e realizado, e trata todos os erros de forma igualitária.

5.4.1 Random Forest

No processo de treinamento do modelo *Random Forest* primeiro importamos a classe *RandomForestRegressor* da biblioteca do *scikit-learn*, que será utilizada tanto para este modelo quanto para as métricas RMSE e MAE que utilizaremos em todos os demais modelos.

O *Random Forest*, por ser um modelo baseado em *ensemble* de árvores de decisão utiliza de alguns hiperparâmetros comuns a esta família de algoritmos. Para que possa ser definida a classe *RandomForestRegressor* é necessário definir o valor desses hiperparâmetros, que serão os valores inicias (ou *baseline*) e que posteriormente terão uma busca por otimização a ser realizada pelo optuna. Os hiperparâmetros selecionados foram os seguintes:

Figura 35 – Hiperparâmetros do *Random Forest*

```
params_rf = {
    'n_estimators': 1000,
    'max_depth': 10,
    'min_samples_split': 10,
    'min_samples_leaf': 10
}

rf_model_g = RandomForestRegressor(**params_rf)
rf_model_e = RandomForestRegressor(**params_rf)
```

n_estimators: Define o número de árvores de decisão que irão compor a “floresta” do modelo. Quanto maior o número de árvores, mais robusto e preciso o modelo tende a ser, pois ele combina os resultados de mais árvores para reduzir o erro de previsão, todavia aumentar o número de árvores aumenta o tempo de treinamento e computação do modelo.

max_depth: Define a “profundidade” de cada árvore que irá compor o modelo. A profundidade da árvore controla o número de divisões (ou nós) que cada árvore pode ter, uma profundidade maior permite que a árvore realize mais divisões nos seus “galhos”, capturando padrões mais complexos e mais informações acerca dos dados. Uma desvantagem de aumentar muito o número do *max_depth* é que o modelo pode ser levado ao *overfitting*, se ajustando demais aos dados e perdendo a capacidade de generalização.

min_samples_split: Define o número mínimo de amostras necessárias para que um nó interno seja dividido em dois novos nós. Aumentar esse parâmetro impede que o modelo crie divisões em nós com poucos dados, o que levaria a um possível sobreajuste, já que criaria a possibilidade da existência de nós nas árvores que contivessem apenas uma instância dos dados, memorizando detalhes apenas daquele único dado. O parâmetro força o modelo a criar divisões apenas quando há dados suficientes, o que ajuda a suavizá-lo modelo e aumenta a capacidade de generalizar para novos dados.

min_samples_leaf: É o parâmetro que define o número mínimo de amostras que deve estar presente em um nó final (denominado nó-folha). Aumentar o *min_samples_leaf* atua de forma semelhante ao parâmetro *min_samples_split* pois faz com que as árvores não contenham no final do seu caminho de decisão poucos dados, o que geraria árvores muito específicas/muito ajustadas aos dados. Além disso, ajustar esse parâmetro pode tornar o modelo mais simples, evitando que regras muito complexas e com difícil interpretação sejam geradas.

O treinamento do modelo é feito em cima dos dados de treino primeiramente para o Etanol e posteriormente para a Gasolina, a partir dos índices criados com o objeto de *cross-validation* para séries temporais (*Time Series Cross-Validation*), utilizando do método *fit()* do objeto do modelo. Depois do treinamento já é possível fazer a avaliação do modelo em dados que ele ainda não viu, ou seja, dados simulando a realidade, os dados de teste. Isso é feito com o método *predict()*. Dessa forma, o modelo é treinado e avaliado em cada *fold* dos dados, de modo que o modelo possa aprender e prever homogeneamente a fim de evitar o *overfitting.*

Figura 36 – Treinamento do modelo *Random Forest* para o dataset do Etanol (idêntico para a Gasolina)

```
## REALIZAÇÃO DO TREINO DO MODELO, PREVISÃO E ARMAZENAMENTO DOS RESULTADOS/SCORES
i=0
rmse_scores_etanol = []
mae_scores_etanol = []
preco_etanol_real = []
preco_etanol_pred = []

### LOOP PARA TREINO E PREDIÇÃO UTILIZANDO CROSS-VALIDATION DE SÉRIES TEMPORAIS
for train_index, test_index in tscv.split(X_etanol):
    X_etanol_train, X_etanol_test = X_etanol.iloc[train_index], X_etanol.iloc[test_index]
    y_etanol_train, y_etanol_test = y_etanol.iloc[train_index], y_etanol.iloc[test_index]

    ### TREINAMENTO DO MODELO
    rf_model_e.fit(X_etanol_train, y_etanol_train)

    ### PREDIÇÃO E CÁLCULO DO RMSE
    y_pred_etanol = rf_model_e.predict(X_etanol_test)
    rmse = root_mean_squared_error(y_etanol_test, y_pred_etanol)
    mae = mean_absolute_error(y_etanol_test, y_pred_etanol)
    print(f"RMSE do FOLD_{i}: {rmse}        ||      MAE do FOLD_{i}: {mae}")
    rmse_scores_etanol.append(rmse)
    mae_scores_etanol.append(mae)

    ### ARMAZENAMENTO DOS RESULTADOS
    preco_etanol_real.extend(y_etanol_test)
    preco_etanol_pred.extend(y_pred_etanol)
    i+=1
```

Após a avaliação do modelo em cada *fold,* sendo que o último deles consiste na totalidade dos dados de treino/teste, é possível obter quais foram os valores das métricas RMSE e MAE para o modelo pela média dessas métricas em cada *fold* dos dados.

Etanol – Random Forest		
	Raiz Erro Q. Médio (RMSE)	**Erro Absoluto Médio (MAE)**
Fold 1	0.080	0.056
Fold 2	0.117	0.080
Fold 3	0.095	0.072
Fold 4	0.072	0.057
Valor final (média)	**0.091**	**0.066**

Gasolina – Random Forest		
Fold 1	0.110	0.081
Fold 2	0.127	0.097
Fold 3	0.131	0.096
Fold 4	0.081	0.065
Valor final (média)	**0.113**	**0.085**

5.4.2 XGBoost

O *XGBoost* é um modelo que também pode ser aplicado à regressão, e a classe utilizada para tal é o *XGBRegressor* da biblioteca própria do modelo, denominada *xgboost* que é importada para obtenção dessa classe.

O modelo emprega, para regressão, árvores de decisão que nesse caso corrigem e buscam melhorar a perfomance das árvores anteriores de forma recursiva e não de forma a somarem suas perfomances para compor um *ensemble* final como é o caso do *Random Forest*. Os hiperparâmetros do *XGBoost* são semelhantes aos do *Random Forest*, com alguns outros a mais. Os hiperparâmetros *default*/iniciais selecionados foram os seguintes:

Figura 37 – Hiperparâmetros do XGBoost

```
## DEFINIÇÃO DOS PARÂMETROS DO MODELO E DO OBJETO DO XGBOOST REGRESSOR
params_xgb = {
    'objective': 'reg:squarederror',
    'eval_metric': 'rmse',
    'learning_rate': 0.01,
    'max_depth': 5,
    'n_estimators': 1000
}

xgb_model_g = XGBRegressor(**params_xgb)
xgb_model_e = XGBRegressor(**params_xgb)
```

objective: Define a função de perda/objetivo do modelo, esta função será a que o modelo buscará minimizar no seu treinamento, utilizando da técnica de *gradiente boosting* como fora explicado na seção de Escolha de modelos. A escolha da função se relaciona com o dado que queremos modelar e com a métrica escolhida. No caso do nosso problema, uma regressão, então escolheu-se como função de perda o erro quadrático, dessa forma o modelo buscará reduzir a diferença quadrática entre o previsto x realizado.

eval_metric: A métrica que o modelo utilizará como avaliação e para monitorar seu progresso e performance em cada iteração do algoritmo de *gradiente boosting*. A escolha deste hiperparâmetro é importante para que o modelo possa identificar se está ocorrendo melhora na sua performance em cada iteração ou não.

learning_rate: Refere-se ao tamanho do passo que o modelo utiliza para ajustar as previsões feitas. Tamanho do passo refere-se ao ajuste que o modelo faz em cada iteração em busca de um mínimo para a função de perda, como essa função pode ter vários mínimos locais o *learning_rate* é um dos ajustes que pode auxiliar para que a função encontre um mínimo global mais rapidamente ou mais lentamente, o que denominamos convergência. Reduzir o *learning_rate* faz com que o modelo dê passos menores a cada iteração, tornando lenta a convergência, mas estando menos sujeito a "errar" o mínimo da função de perda, já para passos maiores, a função pode convergir mais rápido, mas com o risco de perder o mínimo desejado.

O treinamento dos modelos para cada combustível foi feito da mesma forma que para o *Random Forest*, seguindo os *folds* do *cross-validation* para séries temporais de cada combustível. Invoca-se o método *fit()* para o treinamento dos modelos com os dados de treinamento e em seguida o *predict()* para a avaliação do modelo nos dados de teste.

Figura 38 – Treinamento do modelo *XGBoost* para o dataset da Gasolina (Idêntico para o Etanol)

```
## REALIZAÇÃO DO TREINO DO MODELO, PREVISÃO E ARMAZENAMENTO DOS RESULTADOS/SCORES
i=0
rmse_scores_gasolina = []
mae_scores_gasolina = []
preco_gasolina_real = []
preco_gasolina_pred = []

### LOOP PARA TREINO E PREDIÇÃO UTILIZANDO CROSS-VALIDATION DE SÉRIES TEMPORAIS
for train_index, test_index in tscv.split(X_gasolina):
    X_gasolina_train, X_gasolina_test = X_gasolina.iloc[train_index], X_gasolina.iloc[test_index]
    y_gasolina_train, y_gasolina_test = y_gasolina.iloc[train_index], y_gasolina.iloc[test_index]

    ### TREINAMENTO DO MODELO
    xgb_model_g.fit(X_gasolina_train, y_gasolina_train)

    ### PREDIÇÃO E CÁLCULO DO RMSE
    y_pred_gasolina = xgb_model_g.predict(X_gasolina_test)
    rmse = root_mean_squared_error(y_gasolina_test, y_pred_gasolina)
    mae = mean_absolute_error(y_gasolina_test, y_pred_gasolina)
    print(f"RMSE do FOLD_{i}: {rmse}        ||      MAE do FOLD_{i}: {mae}")
    rmse_scores_gasolina.append(rmse)
    mae_scores_gasolina.append(mae)

    ### ARMAZENAMENTO DOS RESULTADOS
    preco_gasolina_real.extend(y_gasolina_test)
    preco_gasolina_pred.extend(y_pred_gasolina)
    i+=1
```

Após a predição ser realizada em cada *fold* dos dados, os valores para as métricas RMSE e MAE foram obtidos, bem como a média geral do modelo para elas.

Etanol - XGBoost		
	Raiz Erro Q. Médio (RMSE)	**Erro Absoluto Médio (MAE)**
Fold 1	0.074	0.058
Fold 2	0.092	0.067
Fold 3	0.114	0.094
Fold 4	0.073	0.058
Valor final (média)	**0.088**	**0.069**

Gasolina - XGBoost		
Fold 1	0.130	0.109
Fold 2	0.128	0.076
Fold 3	0.108	0.088
Fold 4	0.086	0.070
Valor final (média)	**0.113**	**0.086**

5.4.3 SARIMAX

O modelo SARIMA (*Seasonal ARIMA)* é clássico para problema de séries temporais e pode ser utilizado por vias de diversas bibliotecas estatísticas no *Python*, muitas delas que derivam de bibliotecas que foram desenvolvidas na linguagem R, onde muitas implementações de modelos da Estatística começaram de forma computacional, como é o caso do ARIMA/SARIMA.

Neste trabalho, optou-se por utilizar das classes da biblioteca *pmdarima* que implementam a famosa biblioteca *auto.arima* do R em linguagem Python. Com isso, será possível já no processo de treinamento e avaliação do modelo SARIMAX, buscar os melhores hiperparâmetros para os dados deste trabalho, eliminando assim uma etapa posterior de otimização dos parâmetros que será feita com o modelo *XGBoost* e *Random Forest*.

O *auto-arima* é uma classe que busca pelos melhores parâmetros para modelos ARIMA ou SARIMA a partir de um espaço de busca dos hiperparâmetros, nesse espaço de busca são treinados os modelos com os diferentes valores tentativa enquanto se avalia através de testes (Dickey-Fuller por exemplo) e métricas, como AIC e BIC, a performance do modelo, até convergir a um modelo final [15]. Os hiperparâmetros do modelo SARIMAX presentes e que são otimizados são:

Parâmetros não-sazonais (p, d, q): São os parâmetros referentes às componentes do modelo ARIMA padrão, que se referem à tendência e a componente de autocorrelação da série temporal. O **p** representa a ordem do componente autorregressivo (AR), definindo quantos *lags* (períodos passados) da série serão utilizados para previsão do valor atual, é geralmente determinado pelos valores do gráfico PACF. O parâmetro **d** refere-se ao número

de vezes que a série é diferenciada a fim de buscar um comportamento estacionário, isto é, buscando remover a tendência, geralmente não são feitas muitas diferenciações. Por fim o parâmetro **q** refere-se à ordem da componente de médias móveis (MA), definindo quantos resíduos (erros) de períodos passados são relevantes para previsão do valor da variável atual, é geralmente determinado pelos valores do gráfico ACF.

Parâmetros sazonais (P, D, Q, s): Estes são os parâmetros que capturam os padrões de sazonalidade que dá nome ao modelo SARIMA, sem eles o modelo considera que não existe sazonalidade, o que não é o caso para os preços dos combustíveis por exemplos, como foi visto na etapa de análise exploratória. Tanto o **P** quanto o **D** e **Q** são os análogos dos parâmetros autorregressivos, de diferenciação e de médias móveis respectivamente para a sazonalidade da série, existindo pelos mesmos propósitos que os parâmetros (p, d, q) tinham para a parte ARIMA. O parâmetro **s** define o período sazonal que a série possui, ou seja, o ciclo no qual os valores da variável apresentam uma frequência definida, o que para o caso dos preços de combustíveis considera-se uma sazonalidade de 7 dias (semanal).

O treinamento (e otimização) do modelo SARIMAX para os dados de combustíveis é feito com o auxílio da classe *auto_arima* que têm como argumentos os dados de treino e o período sazonal (representado pelo parâmetro m). Os valores iniciais dos hiperparâmetros (p, d, q, P, D, Q) também podem ser passados, mas neste caso serão os valores *default* da classe, visto que a otimização destes por vias de testes e avaliação de critérios de informação irá obter os melhores valores para eles.

A própria classe *auto_arima* realiza o método para treinamento (semelhante ao *fit* para o *XGBoost/Random Forest*), e posteriormente utiliza-se o *predict* passando-se o número de períodos que se deseja prever para avaliação do modelo.

Figura 39 – Treinamento do modelo SARIMAX para o dataset do Etanol (idêntico para a Gasolina)

```
## REALIZAÇÃO DO TREINO DO MODELO, PREVISÃO E ARMAZENAMENTO DOS RESULTADOS/SCORES
i=0
rmse_scores_etanol = []
mae_scores_etanol = []
preco_etanol_real = []
preco_etanol_pred = []

### LOOP PARA TREINO E PREDIÇÃO UTILIZANDO CROSS-VALIDATION DE SÉRIES TEMPORAIS
for train_index, test_index in tscv.split(X_etanol):
    X_etanol_train, X_etanol_test = X_etanol.iloc[train_index], X_etanol.iloc[test_index]
    y_etanol_train, y_etanol_test = y_etanol.iloc[train_index], y_etanol.iloc[test_index]

    ### TREINAMENTO DO MODELO
    best_arima_e = auto_arima(y = y_etanol_train, X = X_etanol_train, m = 7)

    ### PREDIÇÃO E CÁLCULO DO RMSE
    y_pred_etanol = best_arima_e.predict(n_periods = len(X_etanol_test), X = X_etanol_test)
    rmse = root_mean_squared_error(y_etanol_test, y_pred_etanol)
    mae = mean_absolute_error(y_etanol_test, y_pred_etanol)
    print(f"RMSE do FOLD_{i}: {rmse}        ||      MAE do FOLD_{i}: {mae}")
    rmse_scores_etanol.append(rmse)
    mae_scores_etanol.append(mae)

    ### ARMAZENAMENTO DOS RESULTADOS
    preco_etanol_real.extend(y_etanol_test)
    preco_etanol_pred.extend(y_pred_etanol)
    i+=1
```

Após a predição ser realizada em cada *fold* dos dados, os valores para as métricas RMSE e MAE foram obtidos, bem como a média geral do modelo para elas.

Etanol - SARIMAX		
	Raiz Erro Q. Médio (RMSE)	**Erro Absoluto Médio (MAE)**
Fold 1	0.067	0.050
Fold 2	0.070	0.057
Fold 3	0.059	0.046
Fold 4	0.098	0.077
Valor final (média)	**0.074**	**0.057**

Gasolina – SARIMAX		
Fold 1	0.069	0.051
Fold 2	0.081	0.060
Fold 3	0.089	0.067
Fold 4	0.069	0.057
Valor final (média)	**0.077**	**0.058**

5.5. Otimização de hiperparâmetros

A última etapa referente à elaboração dos modelos para prever o preço médio dos combustíveis no estado de MG consiste no chamado *tuning* dos hiperparâmetros dos modelos *Random Forest e XGBoost*. Esse processo é importante de ser realizado com esses modelos visto que os parâmetros definidos no primeiro treinamento foram escolhidos de forma manual, sendo que na otimização é possível avaliar quais seriam as melhores escolhas com base em testes. O modelo SARIMAX não têm a otimização realizada pois esta já foi feita com a construção do mesmo a partir da classe *auto_arima*, que por si só já busca os melhores parâmetros no treinamento.

Na otimização o objetivo principal é ajustar os parâmetros dos modelos para maximizar seu desempenho, considerando neste caso o problema de séries temporais. Esse estudo é crucial, pois permite encontrar a configuração de parâmetros que resulta em um modelo mais preciso e eficiente. Apesar disso essa etapa não substitui uma boa análise de dados e construção de variáveis significativas, sendo tido mais como um "ajuste fino" da performance que o fator mais determinante dela.

O estudo de otimização foi feito com auxílio da biblioteca *optuna*. A biblioteca adota uma abordagem de otimização baseada em ensaios (*trials*), onde diferentes combinações de parâmetros são testadas, e o desempenho de cada configuração é avaliado com base em uma métrica específica, neste caso, o erro médio quadrático (RMSE). O processo de otimização foi implementado em três etapas principais:

Definição do espaço de busca por Hiperparâmetros: O primeiro passo na utilização do Optuna foi definir o espaço de busca, que consiste em definir quais hiperparâmetros que

o algoritmo deve otimizar. Esses parâmetros podem incluir, por exemplo, a profundidade máxima das árvores de decisão (*max_tree_depth*) no *XGBoost*, o número de estimadores (*n_estimators*) no *Random Forest*, ou até mesmo a taxa de aprendizado dos modelos (*learning rate*). Esse espaço de busca foi configurado para ser explorado de forma eficiente pelo Optuna, usando distribuições apropriadas, tais como distribuições **discretas** para parâmetros inteiros e **contínuas** para parâmetros com valores que necessitam de ajuste mais fino (em poucas casas decimais).

Criação da função Objetivo: A função objetivo é uma das principais etapas do processo de otimização. Neste caso, a função objetivo foi definida para treinar o modelo com uma determinada configuração de hiperparâmetros, depois avaliar seu desempenho usando os *folds* de validação cruzada seguindo a ordem temporal e, finalmente, retornar a métrica de avaliação utilizada (RMSE). O Optuna, então, tenta minimizar essa métrica, ou seja, busca reduzir o erro das previsões do modelo. A cada *trial*, o Optuna treina o modelo com uma nova configuração de hiperparâmetros, faz previsões no conjunto de validação e calcula o RMSE para avaliar o quão bem o modelo está performando com aqueles parâmetros.

Algoritmo de Otimização: A biblioteca utiliza uma abordagem baseada em método bayesiano de otimização denominado TPE (*Tree-structured Parzen Estimator*. Em vez de testar todas as combinações de hiperparâmetros de maneira exaustiva (o que poderia se tornar ineficiente), ele faz suposições inteligentes sobre quais conjuntos de hiperparâmetros têm maior probabilidade de melhorar o desempenho do modelo. Isso permite que a otimização ocorra de forma mais rápida e eficiente, testando um número reduzido de configurações, mas ainda assim obtendo bons resultados.

Figura 40 – Definição da função de otimização no Optuna

```python
def objective(trial, X, y, modelo):
    if modelo == 'xgboost':
        params = {
        "objective": "reg:squarederror",
        "n_estimators": trial.suggest_int("n_estimators", 100, 1000, log=True),
        "learning_rate": trial.suggest_float("learning_rate", 1e-3, 0.1, log=True),
        "max_depth": trial.suggest_int("max_depth", 2, 20)
        }

        model = XGBRegressor(**params)

    elif modelo == 'random_forest':
        params = {
        "n_estimators": trial.suggest_int("n_estimators", 100, 1000, log=True),
        "min_samples_leaf": trial.suggest_int("min_samples_leaf", 1, 10),
        "min_samples_split": trial.suggest_int("min_samples_split", 2, 10),
        "max_depth": trial.suggest_int("max_depth", 2, 20)
        }

        model = RandomForestRegressor(**params)

    rmse_opt = []

    for train_index, test_index in tscv.split(X):
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]

        ### TREINAMENTO DO MODELO
        model.fit(X_train, y_train)

        ### PREDIÇÃO E CÁLCULO DO RMSE
        y_pred = model.predict(X_test)
        rmse = root_mean_squared_error(y_test, y_pred)
        rmse_opt.append(rmse)

    return np.mean(rmse_opt)
```

Figura 41 – Definição do estudo de otimização no Optuna

```python
def run_optimize(X, y, modelo, n):
    study = optuna.create_study(direction='minimize')
    study.optimize(lambda trial: objective(trial, X, y, modelo), n_trials=n)

    print(f"Melhores parâmetros para o modelo {modelo}: {study.best_params}")
    print("_____")
    return study.best_params
```

Após a definição da função e do estudo de otimização o que resta é aplicar esse estudo aos modelos e obter dentro dos *trials* o melhor conjunto de hiperparâmetros para o *XGBoost* e o *Random Forest* e em seguida retreinar os modelos com esses parâmetros para buscar melhorar as métricas de avaliação.

Figura 42 – Estudo de otimização dos modelos

```
best_params_rf_e = run_optimize(X_etanol, y_etanol, modelo='random_forest', n=50)
best_params_rf_g = run_optimize(X_gasolina, y_gasolina, modelo='random_forest', n=50)

best_params_xgb_e = run_optimize(X_etanol, y_etanol, modelo='xgboost', n=50)
best_params_xgb_g = run_optimize(X_gasolina, y_gasolina, modelo='xgboost', n=50)
```

Os melhores hiperparâmetros são obtidos com o método *best_params*, para cada combustível e cada modelo.

Figura 43 – Melhores parâmetros obtidos para o *Random Forest* e *XGBoost*

```
Melhores parâmetros para o modelo random_forest: {'n_estimators': 390, 'min_samples_leaf': 2, 'min_samples_split': 3, 'max_depth': 15} Etanol
_____________________________________________________________________
Melhores parâmetros para o modelo random_forest: {'n_estimators': 291, 'min_samples_leaf': 1, 'min_samples_split': 2, 'max_depth': 10} Gasolina
_____________________________________________________________________
Melhores parâmetros para o modelo xgboost: {'n_estimators': 959, 'learning_rate': 0.010191579529354435, 'max_depth': 2} Etanol
_____________________________________________________________________
Melhores parâmetros para o modelo xgboost: {'n_estimators': 129, 'learning_rate': 0.07674476373288527, 'max_depth': 2} Gasolina
```

Com isso o resultado para as métricas após o retreino dos modelos com os parâmetros obtidos no Optuna foram os seguintes:

Etanol		
Modelo	**Raiz Erro Q. Médio (RMSE)**	**Erro Absoluto Médio (MAE)**
Random Forest	**0.089**	**0.067**
XGBoost	**0.079**	**0.061**

Gasolina		
Modelo	**Raiz Erro Q. Médio (RMSE)**	**Erro Absoluto Médio (MAE)**
Random Forest	**0.108**	**0.085**
XGBoost	**0.095**	**0.073**

Comparando com o valor para o RMSE obtido nos treinamentos dos dois modelos sem a otimização de hiperparâmetros é possível notar uma redução no erro de aproximadamente **2% a 4% para o modelo de *Random Forest*** e de **10% a 15% para o modelo *XGBoost***, considerando o Etanol e a Gasolina. Essa redução, seja mais discreta no caso do *Random Forest* ou maior como é para o *XGBoost,* mostra que a etapa de busca por melhores hiperparâmetros é essencial para a melhoria da performance do modelo, mas também revela que os ganhos, por não serem tão significativos, que a escolha inicial de hiperparâmetros não foi feita de forma arbitrária e já forneceu bons modelos.

Com isso, já é possível decidir pelo modelo "campeão" e que mais se adequa ao problema de previsão de preço médio futuro de ambos os combustíveis. Essa escolha se baseia principalmente em qual modelo mais conseguiu minimizar a métrica, mas também em análise por meio de gráficos dos valores previstos e nos resíduos. Juntamente dessa análise é possível também obter interpretações dos modelos, de forma a tornar explicável quais variáveis foram mais importantes e como isso afetou as previsões realizadas. Essa etapa, consolidando a parte final do trabalho, é tema da próxima seção.

6. Apresentação dos Resultados

Após a otimização dos hiperparâmetros para os 3 modelos: SARIMAX (feita durante o treinamento), *Random Forest* e *XGBoost* foi possível obter as métricas finais RMSE e MAE para todos, e por fim compará-las para selecionar qual modelo foi melhor. Cada métrica e modelos foram avaliados separadamente para o *dataset* de Etanol e de Gasolina, para que dessa forma caso modelos diferentes forem melhores em cada base de dados, eles não precisem ser escolhidos para a outra.

Os valores das métricas e seu comparativo entre os modelos estão expressos nas tabelas abaixo.

Etanol

Modelo	Raiz Erro Q. Médio (RMSE)	Erro Absoluto Médio (MAE)
Random Forest	**0.089**	**0.067**
XGBoost	**0.079**	**0.061**
SARIMAX	**0.074**	**0.057**

Gasolina

Modelo	Raiz Erro Q. Médio (RMSE)	Erro Absoluto Médio (MAE)
Random Forest	**0.089**	**0.067**
XGBoost	**0.079**	**0.061**
SARIMAX	**0.077**	**0.058**

O modelo com as melhores métricas e que foi selecionado é então o modelo da família ARIMA, considerando os componentes sazonais e variáveis externas – SARIMAX. Apesar disso o modelo *XGBoost* também apresentou boas métricas, diferindo na casa dos décimos de centavos, visto que a previsão realizada é de um preço em reais, devido a isso a escolha do *XGBoost* para modelar esse tipo de problema também pode ser razoável.

Apesar de ser um erro médio na casa dos centavos, em torno de sete centavos para o modelo SARIMAX, esse resíduo quando considerado no valor praticado nos postos em todo

o estado e no Brasil pode refletir no bolso do cidadão, dessa forma, o próximo passo neste trabalho está em avaliar o modelo selecionado utilizando de gráficos e métricas que permitam o entendimento de como estão a qualidade das previsões, para buscar *insights* e possíveis melhorias nos modelos em trabalhos futuros.

A avaliação dos resultados do modelo selecionado será baseada em três análises gráficas: a primeira refere-se aos erros do modelo e a análise de sua distribuição por meio de um histograma de resíduos, a segunda trata da análise das previsões em comparação aos valores reais e como o desempenho se deu globalmente e por fim qual foram as contribuições das variáveis mais importantes por meio do gráfico de *feature importance*, que foi realizado em meio aos resultados do modelo *XGBoost* por ser uma ferramenta que o modelo dispõe.

Esses gráficos são importantes para avaliar a eficácia do modelo e identificar possíveis melhorias na modelagem.

Figura 44 – Histograma dos resíduos do modelo final para o preço médio do Etanol

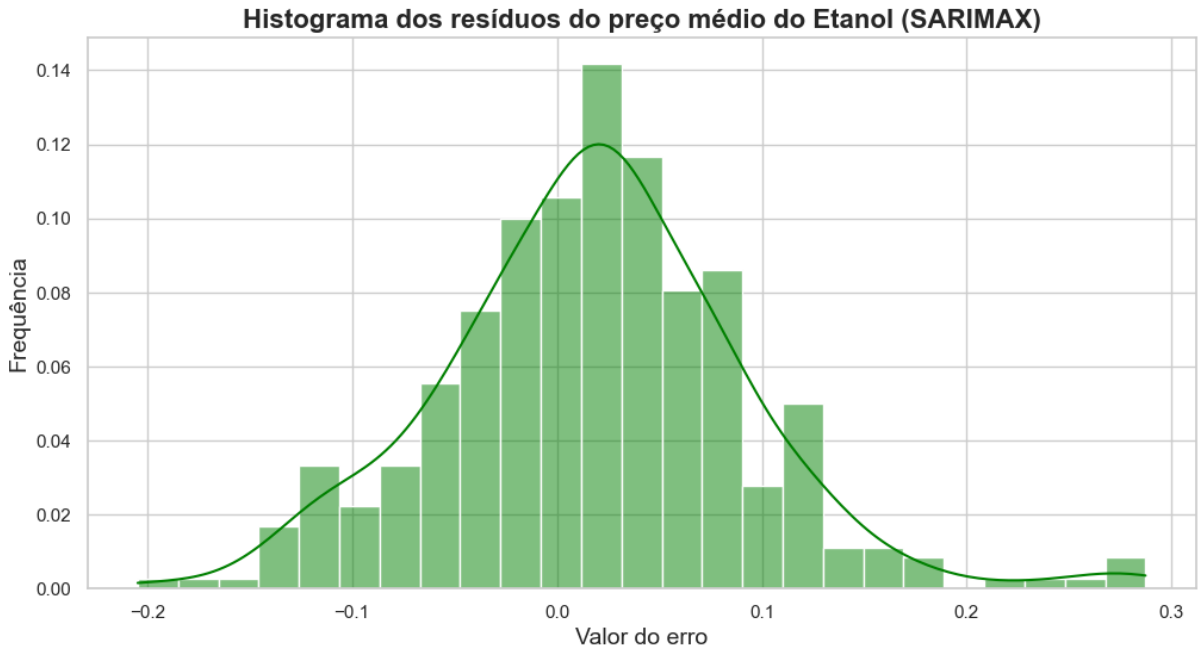
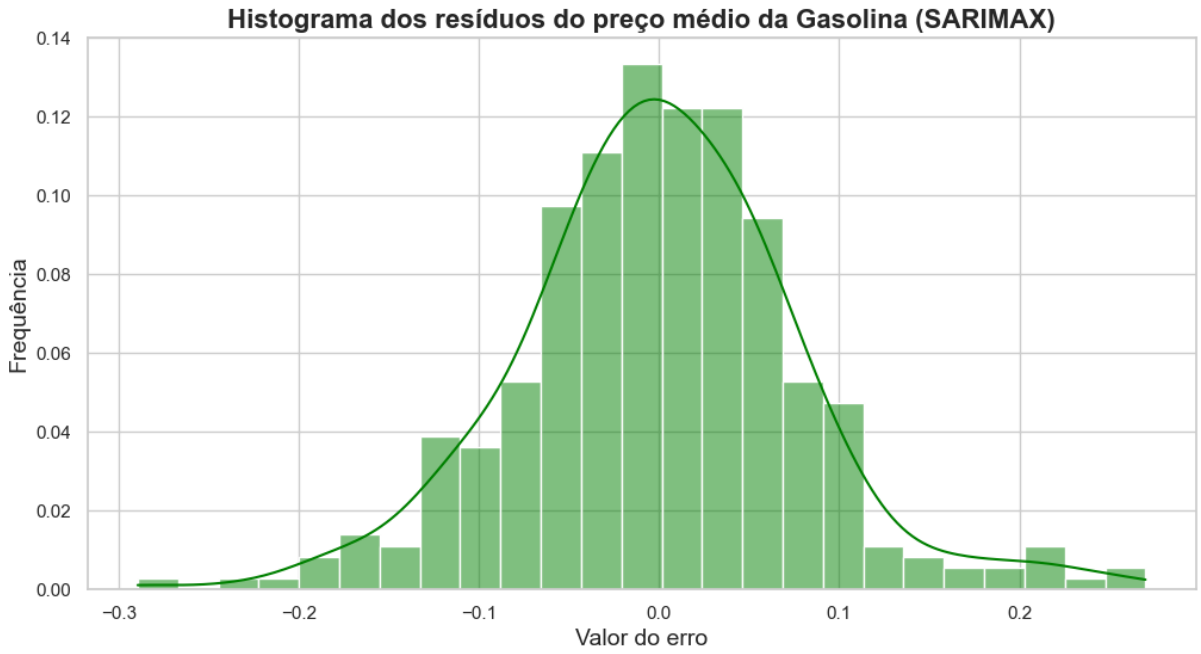


Figura 45 – Histograma dos resíduos do modelo final para o preço médio da Gasolina



Os histogramas dos resíduos, tanto para as previsões do Etanol quanto para da Gasolina fornecem uma visão geral sobre a distribuição dos erros de previsão gerados pelo modelo SARIMAX. No eixo horizontal estão representados os valores dos erros, calculados pela diferença entre o valor previsto e o valor realizado, já no eixo vertical estão expressas as proporções/probabilidades (normalizadas) de ocorrência daquele valor de erro no conjunto de dados que fora previsto pelo modelo. É possível notar, em análise, que ambas as distribuições se aproximam de uma distribuição normal, com um desvio pouco significativo à direita. Isso indica que o modelo selecionado teve êxito em minimizar os erros de previsão, isto é, realizou previsões imprecisas raramente, mas com uma tendência de subestimar alguns valores de previsão (o que é indicado pelos resíduos positivos). O fato de a curva ser aproximadamente simétrica em torno de zero sugere que o modelo, em geral, não apresenta viés significativo, mas as pequenas discrepâncias precisam ser monitoradas, especialmente por se tratar de um contexto envolvendo preços de bens de suma importância para a população, onde variações pequenas no preço podem gerar impactos significativos.

Figura 46 – Gráfico de dispersão dos valores real *versus* previsto para o preço médio do Etanol

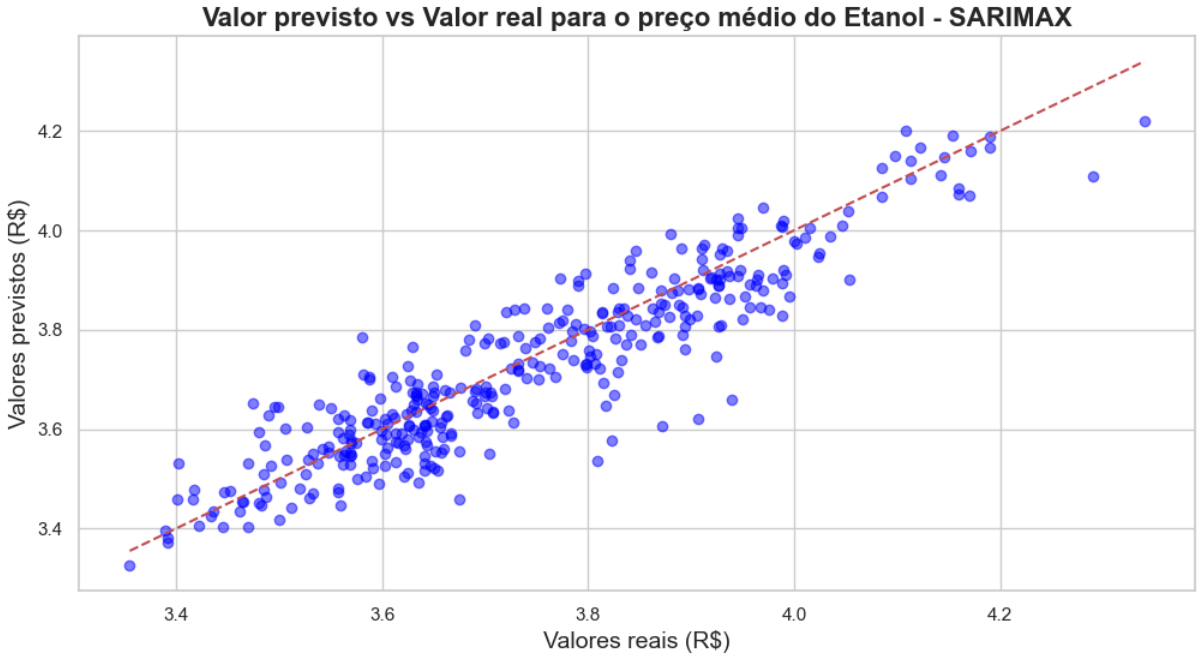
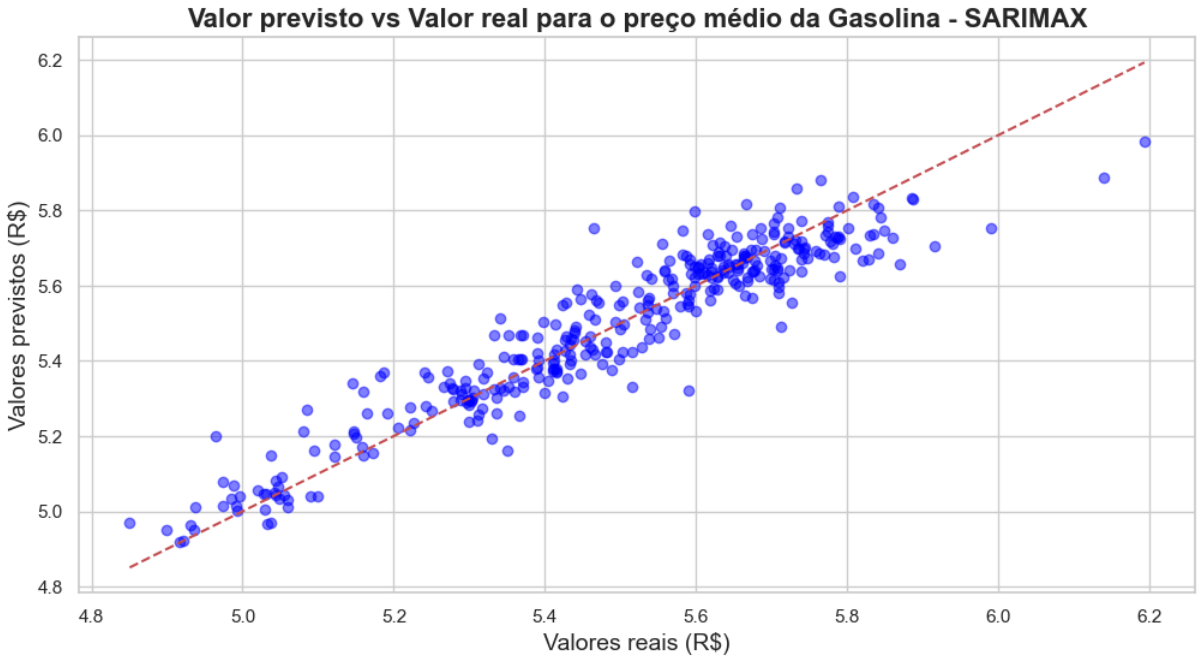


Figura 47 – Gráfico de dispersão dos valores real *versus* previsto para o preço médio da Gasolina



Os gráficos de dispersão acima comparam os valores previstos *versus* valores reais para ambos os combustíveis. No eixo horizontal são colocados os valores reais do preço médio do combustível presente na base de teste (base de avaliação do modelo) e no vertical os valores que o modelo previu para aquela data específica. Também é *plotado* no gráfico uma linha pontilhada de cor vermelha, indicando a linha **ideal de previsão** (y=x), que nos fornece uma base para entender se as previsões estão seguindo o que se considera um bom modelo de regressão.

Apesar de visualmente ser possível acompanhar a tendência dos pontos, no gráfico do Etanol, para valores maiores (preços maiores) há mais dispersão, o que indica que o modelo tem mais dificuldade em prever valores de pico ou que fogem muito do padrão, o que pode ter influência de variáveis externas que não foram incluídas nesse trabalho. Para a Gasolina, a situação é similar. O gráfico mostra um bom *fit* dos valores com a linha de regressão ideal, especialmente nas faixas intermediárias de preço. Entretanto, nos extremos dos preços, o modelo apresenta maior dispersão, sugerindo uma menor precisão nesses casos. Essas discrepâncias em ambos os combustíveis indicam que, apesar do bom desempenho do SARIMAX em termos gerais, há espaço para melhorias, particularmente em prever os extremos da série temporal.

Outras visualizações que auxiliam na análise da qualidade de previsão do modelo selecionado e de como se espera que o modelo mantenha seu comportamento nos próximos períodos é o *plot* da série temporal de cada combustível contemplando todo o período de dados (treino e avaliação), juntamente com as previsões que o modelo realizou.

Figura 48 – Série temporal e previsões do modelo selecionado para o Etanol

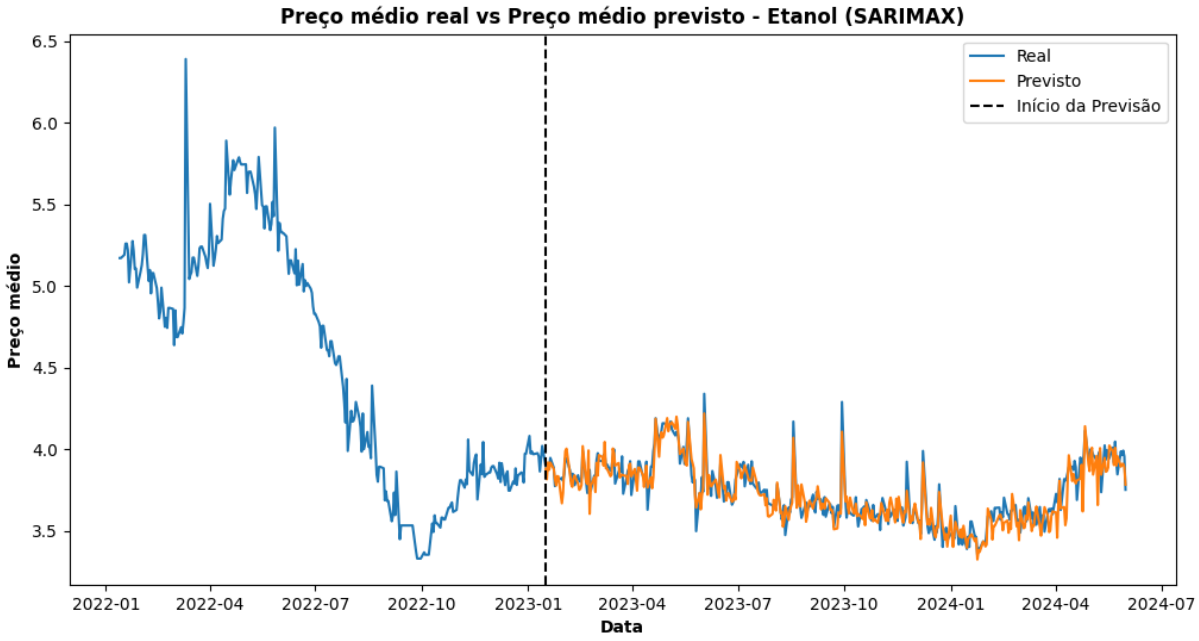
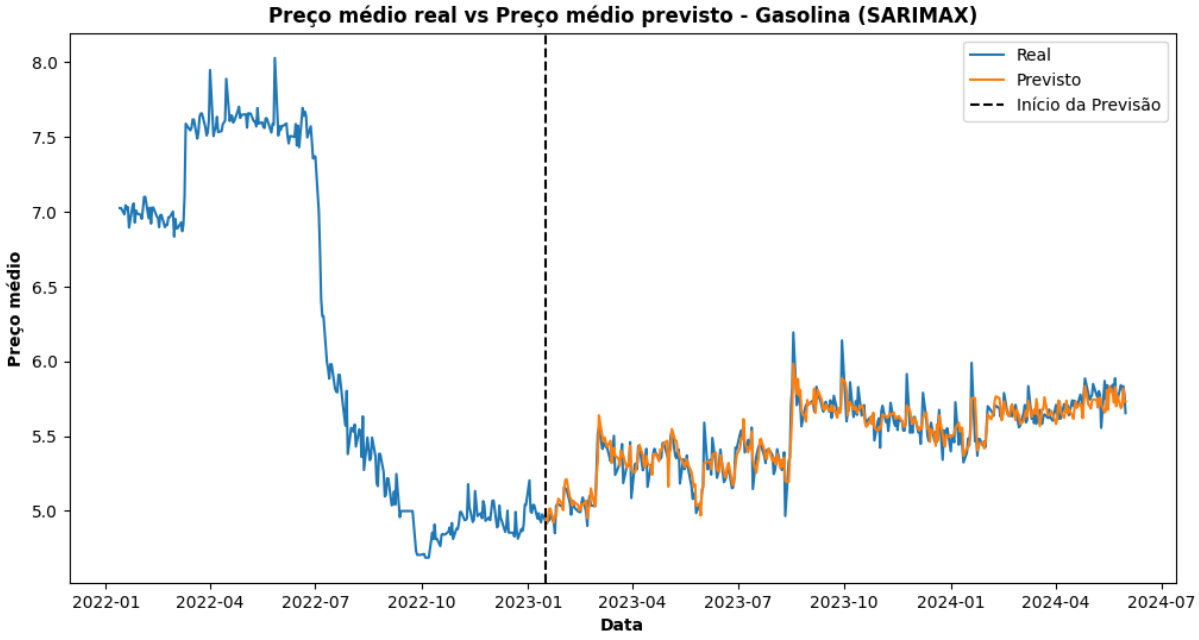


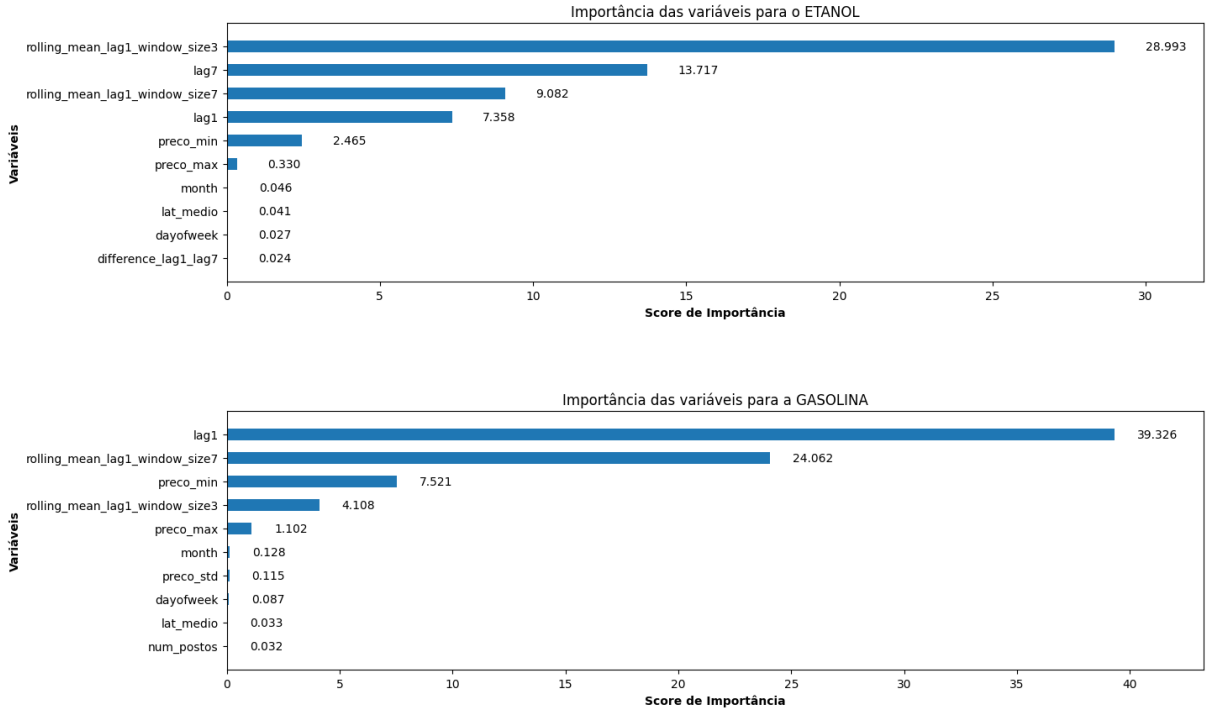
Figura 49 – Série temporal e previsões do modelo selecionado para a Gasolina



Nestes gráficos é possível notar como o modelo SARIMAX aprende sobre certos padrões dos dados tais como a sazonalidade do Etanol e Gasolina nos períodos previstos e picos e vales mais discrepantes. Tal comportamento é possível de ser aprendido pelo modelo

por conta das construções de variáveis temporais como *lags* e médias móveis, essas variáveis dão subsídio para que o modelo autorregressivo possa se atentar a crescimentos ou decrescimentos mais evidentes. Um exemplo da captura desse padrão pode ser vista na variação (crescimento) no preço médio da Gasolina entre os meses 08/2023 à 09/2023 que fica evidente como um pico no gráfico.

Figura 50 – Importância das variáveis fornecidas pelo *XGBoost* para previsão de cada combustível



Os gráficos de importância das variáveis são obtidos a partir do modelo XGBoost, na verdade sendo um gráfico comumente obtido em utilizações de modelos baseados em árvores de decisão, como foi o caso deste trabalho. Apesar de o modelo *XGBoost* não ter sido o modelo com as melhores performance, suas métricas se aproximaram do modelo SARIMAX, devido a isso é razoável considerar a importância das variáveis obtidas nesse modelo para ser adequada ao escopo de todos os modelos testados.

A importância das variáveis de um modelo é uma análise construída com base em um **critério de importância**. Neste caso utilizou-se como critério o **ganho** (*gain*), esse critério considera quais variáveis mais contribuíram para o ganho de *splits* das árvores de decisão, ou seja, quais variáveis foram mais significativas para criar diferentes regras que auxiliaram na previsão em diferentes nós. Outros critérios como o **peso ou *cover*** podem também ser utilizados. A própria biblioteca *XGBoost* fornece uma função para *plotar* as variáveis mais importantes, a *plot_importance()*. No gráfico, o eixo vertical refere-se justamente ao **ganho**,

ou seja, o valor calculado para aquela variável de *score* de importância perante às previsões, no eixo vertical temos as variáveis nomeadas e ordenadas da apresentou maior importância para as previsões em ordem decrescente.

Para o Etanol, é possível notar que as variáveis mais relevantes foram as médias móveis (*rolling mean*) e os *lags* calculado em cima da variável data na etapa de *feature engineering*. Essas variáveis como já esperado são cruciais para capturar a tendência e suavizar flutuações de curto prazo nos dados. O fato de os *lags* também terem alta importância destaca a dependência dos valores passados para prever os preços futuros, algo esperado em séries temporais. Por fim é possível também entender que os valores máximos e mínimos no dia podem influenciar a previsão do preço médio, por diversos fatores como por exemplo eles serem os próprios valores do preço médio (em dias em que só ocorre uma aferição do preço pela pesquisa), ou também por conterem informação sobre as flutuações dos valores.

Para a Gasolina, a estrutura de variáveis é semelhante, com *lags* e médias móveis configurando entre as variáveis com maior importância na previsão. Isso indica que, para ambos os combustíveis, as médias móveis e os valores passados desempenham papéis centrais nas previsões, e que esses valores de curto prazo são mais relevantes que comportamentos de histórico mais longo.

As conclusões que foram possíveis de serem obtidas com o modelo final podem servir como auxílio para tomadas de decisões de organizações públicas ou privadas, pessoas físicas e/ou qualquer ente que dependa do preço dos combustíveis, especialmente os que estão presentes no estado de Minas Gerais, localidade mirada como objetivo deste trabalho. Ser possível fazer um *forecasting* com bom desempenho dos preços médios futuros do Etanol e Gasolina com base em variáveis externas tais como o preço do dólar, variáveis geográficas e afins é uma ferramenta importante para interpretação do resultado dessas coletas de dados por meio de pesquisas do Governo Federal e de entidades contratadas.

Por fim, apesar dos bons resultados obtidos com o modelo da família ARIMA, ainda há espaço para melhorias, futuros estudos podem explorar a inclusão de mais variáveis externas, como por exemplo: dados históricos recentes (retreino do modelo), variáveis externas não consideradas no escopo do trabalho etc. A experimentação com mais modelos, sobretudo modelos de *Deep Learning* pode fornecer resultados mais acurados, considerando que séries temporais é um problema em que arquiteturas de redes neurais como LSTM ou RNN podem fornecer um bom resultado.

8. Links

Link para o vídeo: https://www.youtube.com/watch?v=czjOe6zMJAI

Link para o repositório: https://github.com/emanuel-brandao15/fuel-prices-forecast-brazil-mg

REFERÊNCIAS

[1] Ministério de Minas e Energia – EPE. **Impactos da pandemia de Covid-19 no mercado brasileiro de combustíveis: Reflexos na demanda de combustíveis, na oferta de derivados de petróleo, no setor de biocombustíveis, e análises sobre a arrecadação.** 2020**.** Disponível em: Link. Acesso em 19/12/2023.

[2] InfoMoney. **Corte nos preços de gasolina consolida percepção de IPCA 2023 abaixo do teto da meta.** 2023. Disponível em: Link. Acesso em 19/12/2023.

[3] Portal de Dados Abertos do Governo Federal. **Série Histórica de Preços de Combustíveis e de GLP.** Disponível em: Link. Acesso em 19/12/2023.

[4] Instituto Brasileiro de Petróleo e Gás. **Correlação preço do petróleo e taxa de câmbio.** 2023. Disponível em: Link. Acesso em 04/01/2024.

[5] API-CEP com Base de dados do IBGE. Disponível em: Link. Acesso em 06/05/2024.

[6] GeoJson formatted Brazilian state perimeters. Disponível em: Link. Acesso em 30/04/2024.

[7] Geodata BR – Brasil, perímetro de municípios por Estado. Disponível em: Link. Acesso em 30/04/2024.

[8] O Tempo – Cidades. **Censo 2022: veja ranking das 20 cidades mais populosas de Minas Gerais**. Disponível em: Link. Acesso em 12/05/2024.

[9] Wikipedia – **Kernel density estimation (KDE)**. Disponível em: Link. Acesso em 12/05/2024.

[10] Wikipedia – **Diagrama de Caixa.** Disponível em: Link. Acesso em 14/05/2024.

[11] Kaggle – **Time Series: Interpreting ACF and PACF.** Disponível em: Link. Acesso em 24/05/2024.

[12] Duke University - **Identifying the numbers of AR or MA terms in an ARIMA model.** Disponível em: Link. Acesso em 01/06/2024.

[13] Hyndman, R.J., & Athanasopoulos, G. (2018) *Forecasting: principles and practice*, *second edition*, OTexts: Melbourne, Australia. OTexts.com/fpp2. Acessado em 01/06/2024.

[14] NIXTLA – **MLForecast Library**. Disponível em: Link. Acesso em 14/08/2024.

[15] Autoarima – **Forecast Library (CRAN).** Disponível em: Link. Acesso em 20/08/2024.