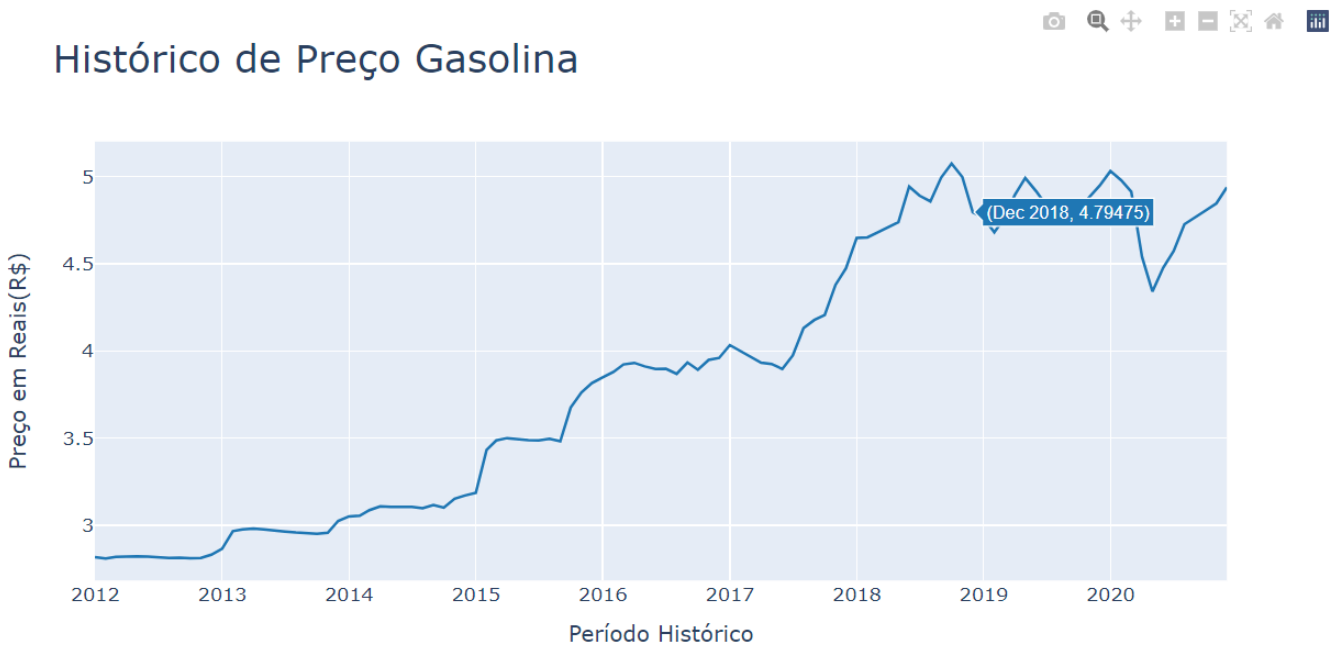
@Odemir Depieri Jr

Prevendo preço da Gasolina



com Serie Temporal

statsmodels

scikit learn

pandas

plotly

matplotlib

Serie Temporal

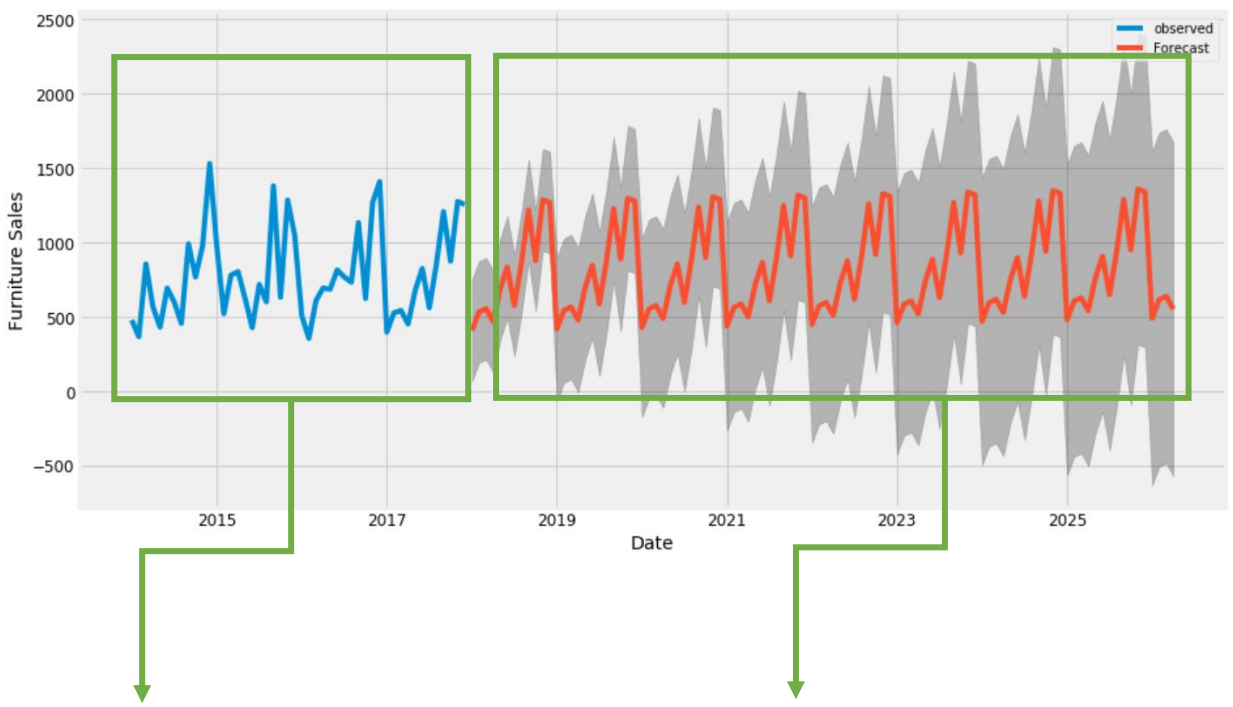
Uma **série temporal** é uma sequência de observações em intervalos de tempo regularmente espaçados.

Exemplo:

- Taxas de desemprego **mensais** para os últimos **cinco anos**
- Produção **diária** em uma fábrica durante um mês
- População em cada **década** de um século

Uma serie temporal **procura padrões** em sequência de intervalos, assim podemos usar a serie temporal para fazer **previsões**.

Exemplo de uma serie gráfica:



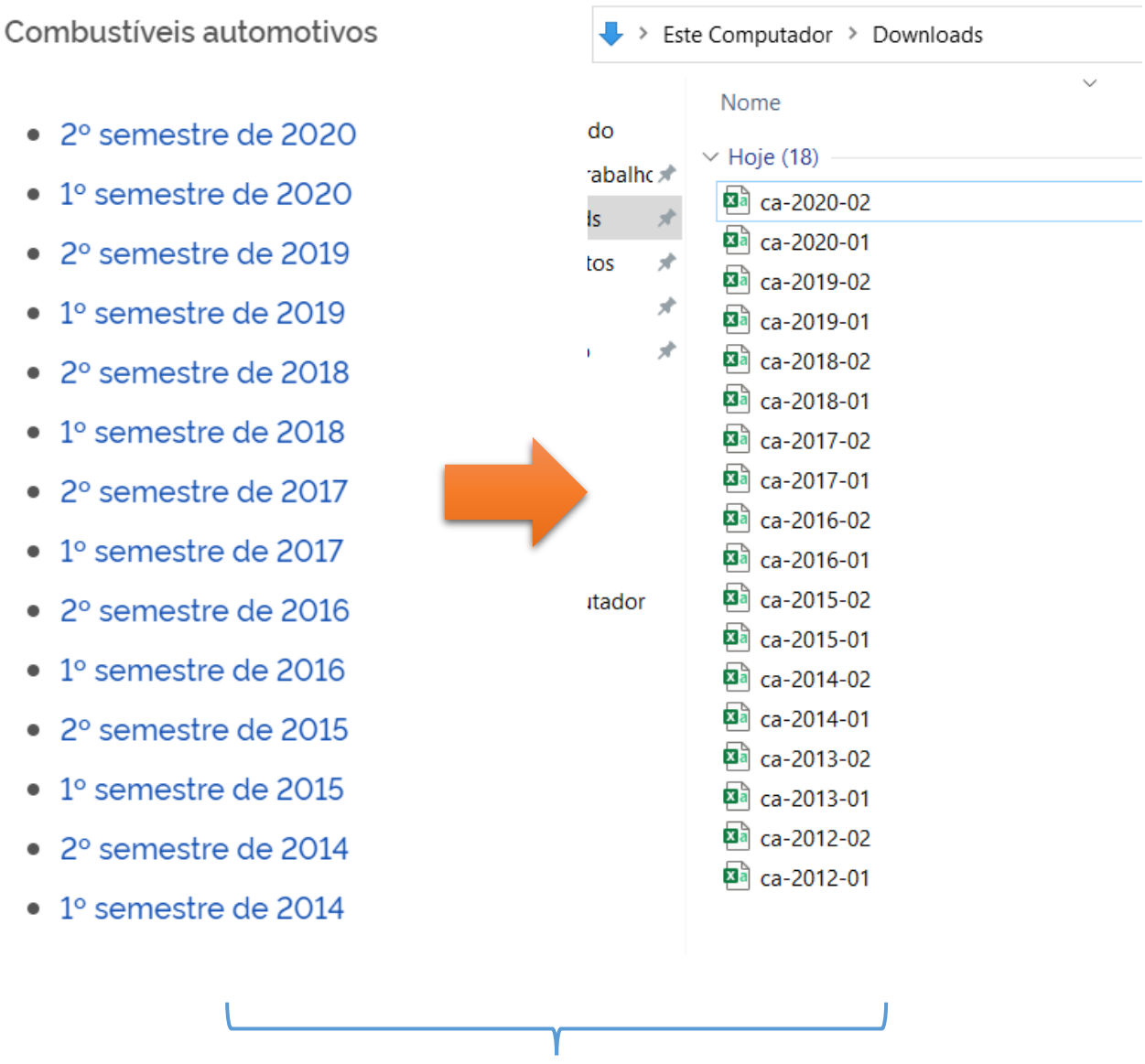
Dados Realizados
(Observação)

Previsão
(Forecast)

Vamos utilizar os dados GOV.BR

https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/serie-historica-de-precos-de-combustiveis

Faça o Download das Bases de Dados



Faça o download de pelo menos de 7 anos para podemos ter uma boa amostra.

Iremos consolidar esses CSV em um único arquivo.

Nesse case vou utilizar o Jupyter Notebook local.

Vamos instalar a Lib 'Ploty'

In [164]:
```
# Instalando a Lib
pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\datav\anaconda3\lib\site-packages (5.2.2)Note: you may need to restart the kernel to use updated packages.
```

Importar nossas Bibliotecas

In [ ]:
```
# Site com os dados da Gasolina
# https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/serie-historica-de-precos-de-combustiveis
```

In [165]:
```
# Função para trabalhar com datas
from datetime import datetime

# Lib para modelagem de Dados
import pandas as pd

# Lib para Analises Gráficas
import matplotlib.pyplot as plt

# Lib para visualização gráfica
import plotly.graph_objects as Dash

# Lib para acessar os recusos do sistema operacional
import os

# Lib para ignorar avisos
import warnings

# Aplicando regra para ignorar avisos
warnings.filterwarnings('ignore')
```

Vamos ler uma base de dados para entendermos como esta seu funcionando.
Mais para frente vamos consolidar todas
Nesse primeiro contato, vamos entender a estrutura de uma base de dados e suas informações

In [63]:
```
# Lendo uma base de dados para entender sua estrutura

# Local para Buscar os arquivos Baixados
Local = 'C:/Users/datav/Downloads/'

# Nome do Arquivo
Arquivo = 'ca-2020-02'

# Entesão
Extensao = '.csv'

# Lendo a base de Dados
Base_Amostra = pd.read_csv( Local + Arquivo + Extensao, sep=';')

# Verificando primeiras linhas
Base_Amostra.head()
```

Out[63]:

	Regiao - Sigla	Estado - Sigla	Municipio	Revenda	CNPJ da Revenda	Nome da Rua	Numero Rua	Complemento	Bairro	Cep	Produto	Data da Coleta	Valor de Venda	Val Comp
0	SE	SP	SOROCABA	COMPETRO COMERCIO E DISTRIBUICAO DE DERIVADOS ...	00.003.188/0001-21	RUA HUMBERTO DE CAMPOS	306	NaN	JARDIM ZULMIRA	18061-000	GASOLINA	01/07/2020	3,559	3,2
1	SE	SP	SOROCABA	COMPETRO COMERCIO E DISTRIBUICAO DE DERIVADOS ...	00.003.188/0001-21	RUA HUMBERTO DE CAMPOS	306	NaN	JARDIM ZULMIRA	18061-000	ETANOL	01/07/2020	2,329	2,11
2	SE	SP	SOROCABA	COMPETRO COMERCIO E DISTRIBUICAO DE DERIVADOS ...	00.003.188/0001-21	RUA HUMBERTO DE CAMPOS	306	NaN	JARDIM ZULMIRA	18061-000	DIESEL S10	01/07/2020	2,859	Na
3	NE	BA	SALVADOR	PETROBRAS DISTRIBUIDORA S.A.	34.274.233/0015-08	RUA EDISTIO PONDE	474	NaN	STIEP	41770-395	GASOLINA	02/07/2020	4,29	3,76
4	NE	BA	SALVADOR	PETROBRAS DISTRIBUIDORA S.A.	34.274.233/0015-08	RUA EDISTIO PONDE	474	NaN	STIEP	41770-395	ETANOL	02/07/2020	3,29	2,66

Filtrando uma região e analisando

In [106]:
```
# Gerando algumas Analises

# Filtrar a Região de SP para nosso estudo
# Produto Gasolina
Filtro_01 = Base_Amostra.loc[ ( Base_Amostra['Produto'] == 'GASOLINA' ) &
                              ( Base_Amostra['Estado - Sigla'] == 'SP' )
                            ]

# Verificando a Dimensão
print('Dimensão da Base de Dados:', '\n', Filtro_01.shape, '\n' )

# Verificando
print( Filtro_01.info() )
```

```
Dimensão da Base de Dados:
 (17952, 16)
```

```
Dimensão da Base de Dados:
 (17952, 16)

<class 'pandas.core.frame.DataFrame'>
Int64Index: 17952 entries, 0 to 222629
Data columns (total 16 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Regiao - Sigla     17952 non-null  object
 1   Estado - Sigla     17952 non-null  object
 2   Municipio          17952 non-null  object
 3   Revenda            17952 non-null  object
 4   CNPJ da Revenda    17952 non-null  object
 5   Nome da Rua        17952 non-null  object
 6   Numero Rua         17952 non-null  object
 7   Complemento        2137 non-null   object
 8   Bairro             17907 non-null  object
 9   Cep                17952 non-null  object
 10  Produto            17952 non-null  object
 11  Data da Coleta     17952 non-null  object
 12  Valor de Venda     17952 non-null  object
 13  Valor de Compra    3089 non-null   object
 14  Unidade de Medida  17952 non-null  object
 15  Bandeira           17952 non-null  object
dtypes: object(16)
memory usage: 2.3+ MB
None
```

In [85]:
```
# Verificando campos nulos
Filtro_01.isnull().sum()
```

Out[85]:
```
Regiao - Sigla          0
Estado - Sigla          0
Municipio               0
Revenda                 0
CNPJ da Revenda         0
Nome da Rua             0
Numero Rua              0
Complemento         15815
Bairro                 45
Cep                     0
Produto                 0
Data da Coleta          0
Valor de Venda          0
Valor de Compra     14863
Unidade de Medida       0
Bandeira                0
dtype: int64
```

Tratamento de Dados

In [86]:
```
# Retirando 'virgula' e incluindo o '.'
Filtro_01['Valor de Venda'] = Filtro_01['Valor de Venda'].str.replace(',', '.', regex=True)

# Convertendo para valor flutuante
Filtro_01['Valor de Venda'] = pd.to_numeric( Filtro_01['Valor de Venda'] )

# Analisando estatisticas do campo
Filtro_01.describe()
```

Out[86]:

	Valor de Venda
count	17952.000000
mean	4.050566
std	0.224615
min	3.259000
25%	3.899000
50%	3.999000
75%	4.199000
max	4.999000

In [97]:
```
# Convertendo a coluna para formato de Data
Filtro_01['Data da Coleta'] = pd.to_datetime( Filtro_01['Data da Coleta'], format="%d/%m/%Y" )

# Gerando o mês
Filtro_01['Mes'] = pd.DatetimeIndex( Filtro_01['Data da Coleta'] ).month

# Gerando o Ano
Filtro_01['Ano'] = pd.DatetimeIndex( Filtro_01['Data da Coleta'] ).year
```

In [104]:
```
# Gerar a Média do Estado para nosso Estudo
Analise = Filtro_01.groupby(['Ano', 'Mes']).mean().reset_index()

# Analise Final
Analise
```

Out[104]:

	Ano	Mes	Valor de Venda
0	2020	7	3.941447
1	2020	8	4.017643
2	2020	10	4.200923
3	2020	11	4.193360
4	2020	12	4.223971

Agora vamos consolidar todas os arquivos e aplicar os mesmo conceitos

In [110]:
```
# ------------ Estrutura para consolidar os arquivo Baixados ------------ #

# Local para Buscar os arquivos Baixados
Local = 'C:/Users/datav/Downloads/'

# Variavel de apoio no Loop
Quantidade_Arquivos = 0

# Loop para varrer o Diretorio de Download
for Diretorio, Subpastas, Arquivos in os.walk(Local):

    # Loop nos arquivos
    for Arquivo in Arquivos:

        # Caso o arquivo começe com 'ca-'
        if 'ca-' in str(Arquivo):

            # Apontando para o arquivo
            Arquivo_Donwload = os.path.join(Diretorio, Arquivo)

            # Lendo a BAse de Dados
            Arquivo_Atual = pd.read_csv( Arquivo_Donwload, sep=';' )

            # caso seja a primeira interação, irá criar o arquivo consolidado
            if Quantidade_Arquivos == 0:
                Base_Consolidada = Arquivo_Atual

            # caso contrario consolida
            else:
                Base_Consolidada = pd.concat([ Base_Consolidada, Arquivo_Atual ])

            # Somando a interação
            Quantidade_Arquivos += 1

print('Foi consolidado', Quantidade_Arquivos, 'Arquivos')
```

```
Foi consolidado 18 Arquivos
```

In [112]:
```
# Verificando a Dimensão da Base de Dados Consolidada
Base_Consolidada.shape
```

Out[112]: (9453671, 16)

Nosso script fez a consolidação de todos os arquivos da pasta Download. Há mais de 9 milhões de registros.

Agora vamos aplicar as mesos conceitos da fase da exploração.

Vamos trabalhar com uma região (Cidade) a Simplício para ter uma melhor precisão.

Aqui você pode escolher qualquer cidade para gerar a previsão.

Filtrando a Cidade do Rio de Janeiro para nosso modelo

In [277]:
```
# Filtrando apenas a região que queremos trabalhar
Filtro_Geral = Base_Consolidada.loc[
    ( Base_Consolidada['Produto'] == 'GASOLINA' ) &
    ( Base_Consolidada['Estado - Sigla'] == 'RJ' ) &
    ( Base_Consolidada['Municipio'] == 'RIO DE JANEIRO')
]

# Nova dimensão
Filtro_Geral.shape
```

Out[277]: (52798, 16)

In [278]:
```
# Vamos pegar as colunas que precisamos para economizar tempo de processamento
Filtro_Geral = Filtro_Geral[['Data da Coleta', 'Valor de Venda']]

# Nova dimensão
Filtro_Geral.shape
```

Out[278]: (52798, 2)

In [279]:
```
# Vamos fazer o tratamento nos dados igual ao exemplo anterior

# Retirando 'virgula' e incluindo o '.'
Filtro_Geral['Valor de Venda'] = Filtro_Geral['Valor de Venda'].str.replace(',', '.', regex=True)

# Convertendo para valor flutuante
Filtro_Geral['Valor de Venda'] = pd.to_numeric( Filtro_Geral['Valor de Venda'] )

# Convertendo a coluna para formato de Data
Filtro_Geral['Data da Coleta'] = pd.to_datetime( Filtro_Geral['Data da Coleta'], format="%d/%m/%Y" )

# Gerando o mês
Filtro_Geral['Mes'] = pd.DatetimeIndex( Filtro_Geral['Data da Coleta'] ).month

# Gerando o Ano
Filtro_Geral['Ano'] = pd.DatetimeIndex( Filtro_Geral['Data da Coleta'] ).year

# Verificando a Base
Filtro_Geral.head()
```

Out[279]:

	Data da Coleta	Valor de Venda	Mes	Ano
7759	2012-01-03	2.799	1	2012
7763	2012-01-03	2.699	1	2012
7767	2012-01-04	2.749	1	2012
7770	2012-01-04	2.899	1	2012
7773	2012-01-03	2.849	1	2012

In [280]:
```
# Gerando nossa Analise

# Gerar a Média do Estado para nosso Estudo
Analise_Geral = Filtro_Geral.groupby(['Ano', 'Mes']).mean().reset_index()

# Analise Final
Analise_Geral
```

Out[280]:

	Ano	Mes	Valor de Venda
0	2012	1	2.818623
1	2012	2	2.810398
2	2012	3	2.820107
3	2012	4	2.818450
4	2012	5	2.822450
...	...	...	...
101	2020	7	4.572690
102	2020	8	4.725896
103	2020	10	4.805013
104	2020	11	4.845562
105	2020	12	4.937116

106 rows × 3 columns

In [281]:
```
# Gerar Data para ser usadas em nossa base de dados

# Criando função para gerar a data
def Gerar_Data(Ano, Mes):

    # Todas as datas serão dia 1
    Dia = 1

    # Função da 'datetime' para criar uma data
    Data = datetime( int(Ano), int(Mes), Dia)

    # Retornando a Data
    return Data

# Aplicando a Função e passando 2 arugmentos dentro do aplly do Pandas
Analise_Geral['Data'] = Analise_Geral.apply( lambda Coluna: Gerar_Data( Coluna['Ano'],Coluna['Mes'] ), axis=1 )

# Verificando primeiras linhas
Analise_Geral.head()
```

Out[281]:

	Ano	Mes	Valor de Venda	Data
0	2012	1	2.818623	2012-01-01

@Odemir Depieri Jr

In [282]:
```
# Plotar os Gráficos
Analise_Grafica = Analise_Geral.set_index('Data')
```

In [283]:
```
Analise_Grafica
```

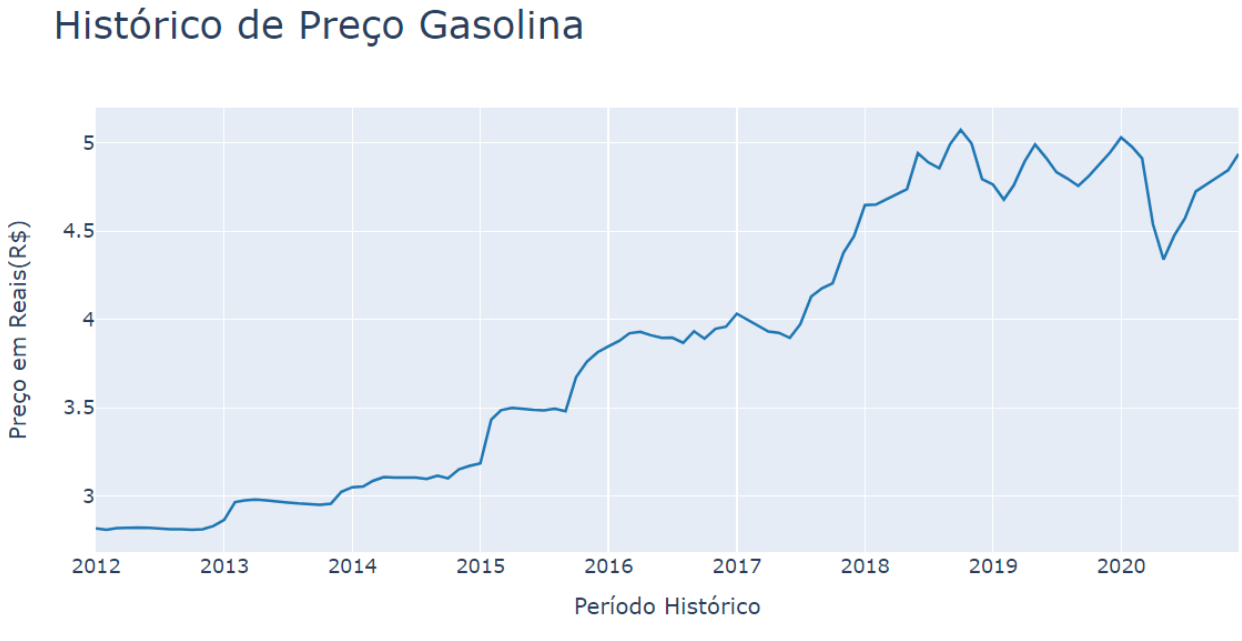
Out[283]:

Data	Ano	Mes	Valor de Venda
2012-01-01	2012	1	2.818623
2012-02-01	2012	2	2.810398
2012-03-01	2012	3	2.820107
2012-04-01	2012	4	2.818450
2012-05-01	2012	5	2.822450
...	...	...	...
2020-07-01	2020	7	4.572690
2020-08-01	2020	8	4.725896
2020-10-01	2020	10	4.805013
2020-11-01	2020	11	4.845562
2020-12-01	2020	12	4.937116

106 rows × 3 columns

In [284]:
```
# Renomenado a Coluna valor de venda
Analise_Grafica.rename( columns={'Valor de Venda':'Valor_Venda'}, inplace=True )
```

In [285]:
```
# Criando um Gráfico Dinâmico
# No gráfico é possivel filtrar pela legenda a informação
# Utlizar zoons

# Definindo uma figura
Figura = Dash.Figure()

# Incluindo o Eixo no Gráfico - Abertura
Figura.add_trace( Dash.Scatter(x = Analise_Grafica.index, y = Analise_Grafica.Valor_Venda,
                    mode='lines',
                    name='Preço',
                    marker_color = '#1F77B4',))


# Modificando o Layout do Gráfico
Figura.update_layout(
    title='Histórico de Preço Gasolina', # Titulo
    titlefont_size = 28, # Tamanho da Fonte

    # Parametros para mexer no eixo X
    xaxis = dict(
        title='Período Histórico', # Titulo do Eixo x
        titlefont_size=16, # Tamanho fonte do Titulo
        tickfont_size=14), # Tamanho da fonte do eixo

    # Tamanho do Grafico
    height = 500,

    # Parametros para mexer no eixo y
    yaxis=dict(
        title='Preço em Reais(R$)', # Titulo do Eixo y
        titlefont_size=16, # Tamanho fonte do Titulo
        tickfont_size=14), # Tamanho da fonte do eixo

    # Parametros para mexer na legenda
    legend=dict(
        y=1, x=1, # Posição da Legenda
        bgcolor='rgba(255, 255, 255, 0)', # Cor de fundo
        bordercolor='rgba(255, 255, 255, 0)')) # Cor da Bornda

# Mostrando o Gráfico
Figura.show()
```



Treinando o Modelo

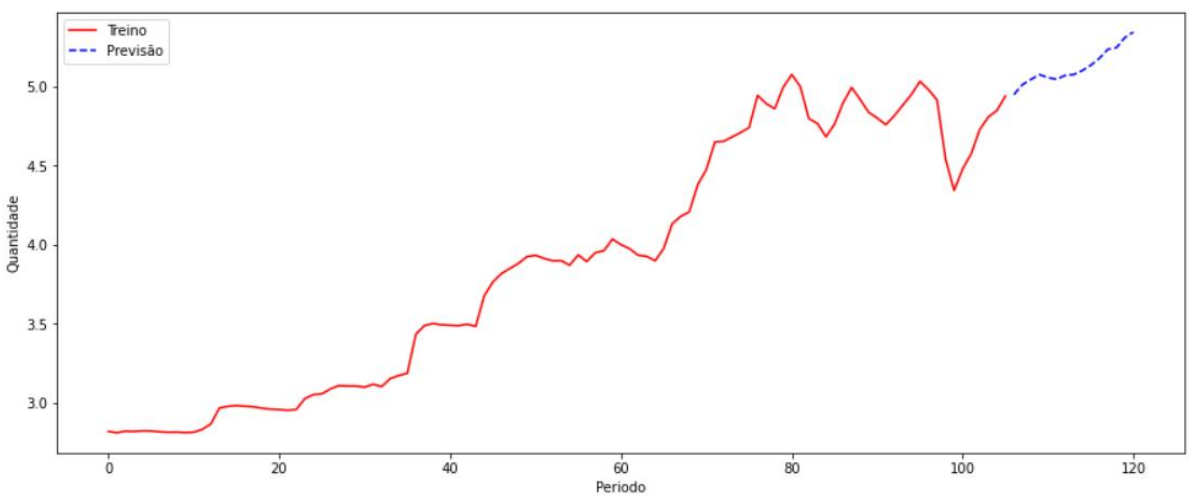
In [308]:
```
# Treinar o modelo da Serie Temporal

Dados_Serie = Analise_Grafica[['Valor_Venda']]

# Importando a Função da Serie
from statsmodels.tsa.api import ExponentialSmoothing

# Definindo os parametros
Funcao_Serie_Temporal = ExponentialSmoothing(
    Dados_Serie,
    seasonal_periods=12,
    trend='additive',
    seasonal='additive').fit(use_boxcox=True)
```

```
C:\Users\datav\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: ValueWarning:

A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
```

In [312]:
```
# Defiindo os dias para serem previstos
Periodos_Para_Prever = 15

# Fazendo a previsao usando o metodo 'FORECAST'
Previsao = Funcao_Serie_Temporal.forecast( Periodos_Para_Prever )
```

```
C:\Users\datav\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:376: ValueWarning:

No supported index is available. Prediction results will be given with an integer index beginning at `start`.
```

In [313]:
```
# Criando o gráfico com a previsão e o realizado

# Definindo o tamanho do Gráfico
plt.figure(figsize=(15,6))

# Plotando os valores reais
plt.plot( Dados_Serie['Valor_Venda'].values, label='Treino', color='red')

# Plotando os dados de Previsão
plt.plot( Previsao, label='Previsão', color='blue', linestyle='--')

# Definindo os Eixos x e y
plt.xlabel('Periodo')
plt.ylabel('Quantidade')
# Posição da Legenda
plt.legend(loc=0);
```



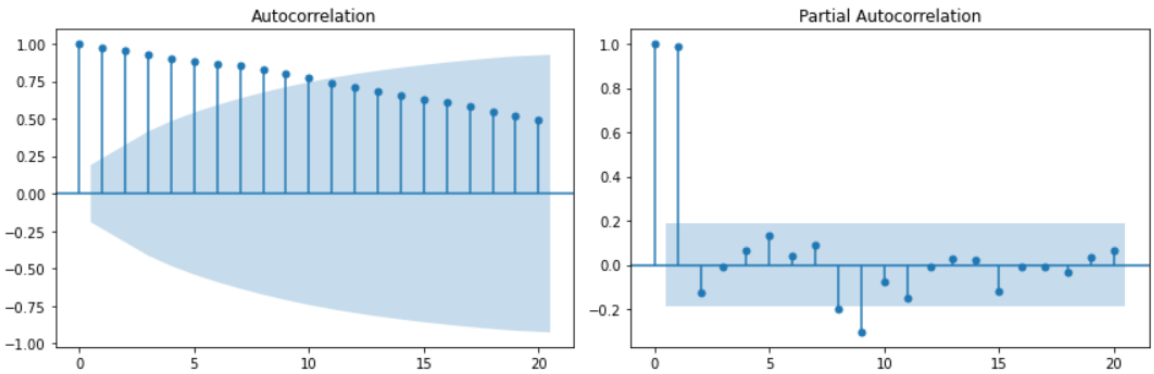
E por fim esta nossa previsão do preço da Gasolina nos próximos meses.

Aqui podemos escolher quantos períodos queremos fazer a previsão.

Veja que entre Abril/20 a Ago/20 houve queda no preço da Gasolina e isso interferiu na previsão do ano de 2021.

Analisando o modelo

In [314]:
```
# Verificando o diagrama de Autocorrelação
# Verificando o diagrama de Autocorrelação parcial

# Importando a função smt para gerar as correlações
import statsmodels.tsa.api as smt

# Definindo uma figura de 1 linha e 2 colunas
fig, axes = plt.subplots(1, 2)

# Fixando o tamanho dos gráficos
fig.set_figwidth(12)
fig.set_figheight(4)

# Plotando o gráfico de AutoCorrelação
smt.graphics.plot_acf(Dados_Serie, lags=20, ax=axes[0])

# Plotando o gráfico de AutoCorrelação PArcial
smt.graphics.plot_pacf(Dados_Serie, lags=20, ax=axes[1])
plt.tight_layout()
```



Caso tenha dúvidas sobre o assunto, recomendo esse vídeo.

https://www.youtube.com/watch?v=JuG8hwVK5uQ

Link da Documentação
https://www.statsmodels.org/stable/index.html

Uruuu !

Final

Esse guia foi elaborada para demostrar como prever preços usando uma serie temporal.

Link da Documentação
https://www.statsmodels.org/stable/index.html

Odemir Depieri Jr

Data Intelligence Analyst Sr
Tech Lead
Specialization AI