Previsão do IPCA

February 14, 2021

1 Análise e estimativa da inflação brasileira

Neste exercício iremos fazer uma análise da inflação brasileira. Os dados são coletados o índice de séries históricas do Banco Central do Brasil e sua periodicidade é mensal. Para uma análise mais clara, a série começara em junho de 1999, onde o país adotou o regime de metas de inflação.

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns; sns.set()
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['figure.figsize']=(16,8)
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima_model import ARIMA
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
import pmdarima as pm
from datetime import date
from dateutil.relativedelta import relativedelta
```

1.0.1 Importando e tratando os dados

https://www3.bcb.gov.br/sgspub/localizarseries/localizarSeries.do?method=prepararTelaLocalizarSeries

Nessa seção é extraido os valores do ipca usando a API do gerador de séries temporais do Bacen. E transformaremos o formato de arquivo json para um dataframe.

[2]:
```
codigo_bcb = 433 #IPCA

def consulta_bcb(codigo_bcb):
    url = 'http://api.bcb.gov.br/dados/serie/bcdata.sgs.{}/dados?formato=json'.
 ↪format(codigo_bcb)
    df = pd.read_json(url)
    df['data'] = pd.to_datetime(df['data'], dayfirst = True).dt.date
    df.set_index('data', inplace = True)

    return df
```

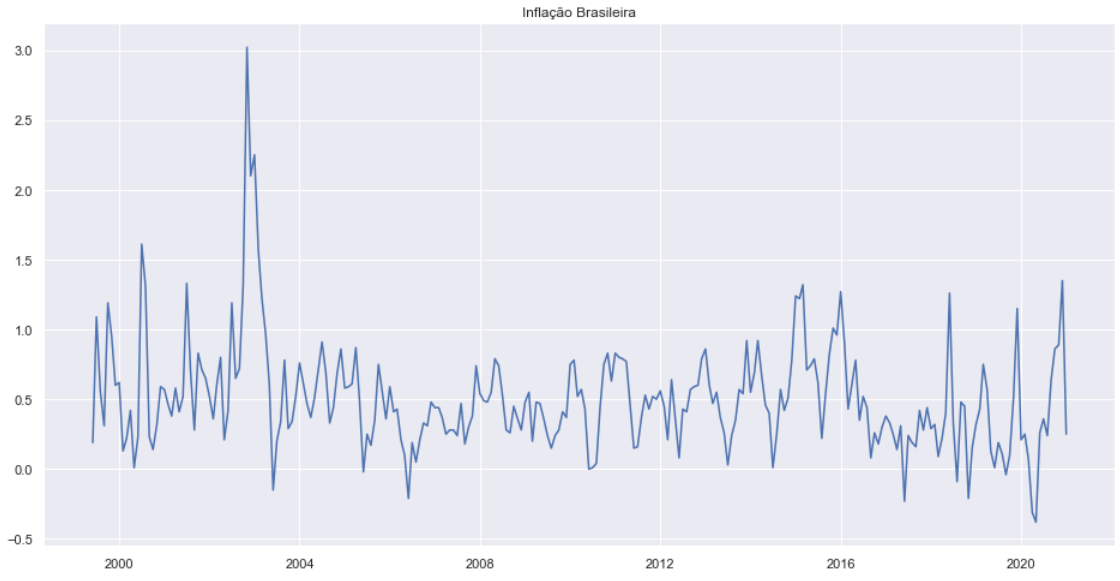
[3]:
```
data_inicio = pd.to_datetime('1999-06-01')
ipca = consulta_bcb(433)
inflacao = ipca[ipca.index >= data_inicio]
inflacao
```

[3]:

data	valor
1999-06-01	0.19
1999-07-01	1.09
1999-08-01	0.56
1999-09-01	0.31
1999-10-01	1.19
…	…
2020-09-01	0.64
2020-10-01	0.86
2020-11-01	0.89
2020-12-01	1.35
2021-01-01	0.25

[260 rows x 1 columns]

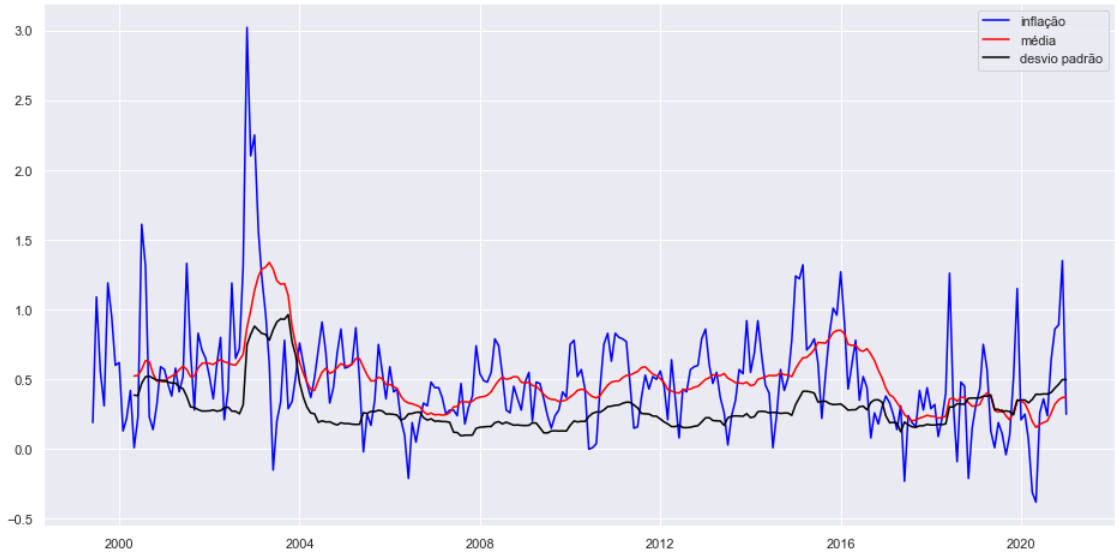
[4]:
```
plt.plot(inflacao)
plt.title('Inflação Brasileira')
plt.show()
```



1.0.2 Conferindo se a série é estacionária

Análise da média e desvio padrão A série temporal é estacionária se permanecer constante com o tempo (a olho nu, veja se as linhas são retas e paralelas ao eixo x).

[5]:
```
rolling_mean = inflacao.rolling(window = 12).mean()
rolling_std = inflacao.rolling(window = 12).std()

plt.plot(inflacao, color = 'blue', label = 'inflação')
plt.plot(rolling_mean, color = 'red', label = 'média')
plt.plot(rolling_std, color = 'black', label = 'desvio padrão')
plt.legend(loc = 'best')
plt.show()
```



Teste dickey-fuller Aumentado A série temporal é considerada estacionária se o valor p for baixo (de acordo com a hipótese nula) e os valores críticos em intervalos de confiança de 1%, 5%, 10% forem o mais próximo possível das Estatísticas ADF

[6]:
```
teste = adfuller(inflacao['valor'])
print('Estatística ADF:{}'.format(teste[0]))
print('P-valor:{}'.format(teste[1]))
print('Valores críticos: {}'.format(teste[4]))
```

```
Estatística ADF:-3.584746166993059
P-valor:0.006059434480841773
Valores críticos: {'1%': -3.4566744514553016, '5%': -2.8731248767783426, '10%':
-2.5729436702592023}
```

Como podemos ver o p-valor é menor que 0,05 ($< 0{,}05$), além de que a média de 12 períodos é constante em relação ao tempo. Portanto a série é estacionária.

1.0.3 Estimação

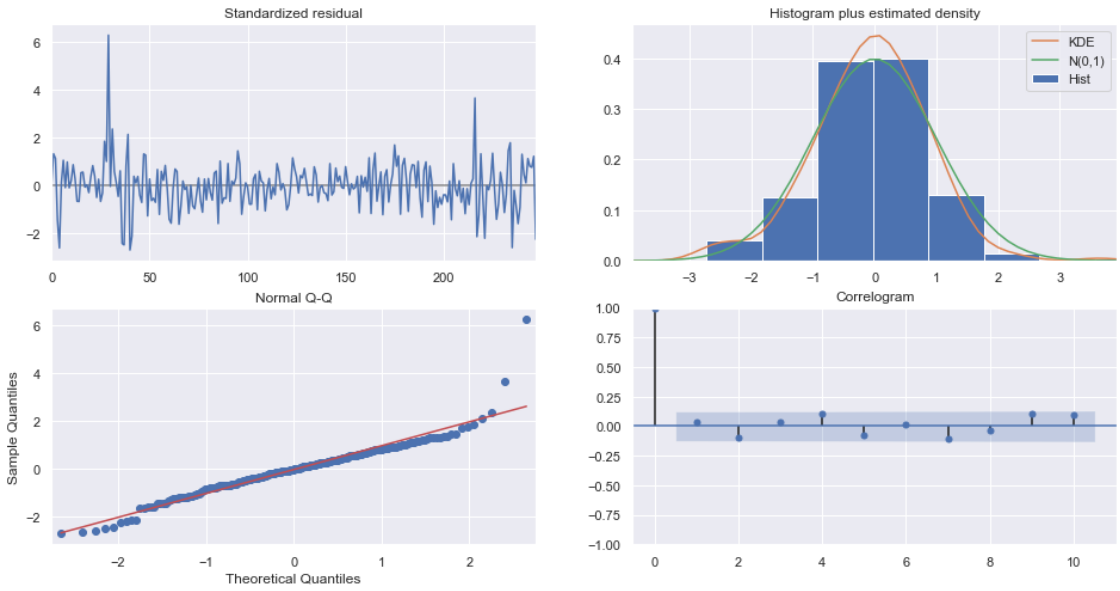
[7]:
```
model = pm.auto_arima(inflacao.valor, start_p=1, start_q=1,
                      test='adf',       # use adftest to find optimal 'd'
                      max_p=3, max_q=3, # maximum p and q
                      m=12,              # frequency of series
                      d=None,           # let model determine 'd'
                      seasonal=True,
                      start_P=0,
                      D=1,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)

print(model.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(0,1,1)[12] intercept   : AIC=128.871, Time=0.70 sec
 ARIMA(0,0,0)(0,1,0)[12] intercept   : AIC=354.477, Time=0.09 sec
 ARIMA(1,0,0)(1,1,0)[12] intercept   : AIC=170.346, Time=0.44 sec
 ARIMA(0,0,1)(0,1,1)[12] intercept   : AIC=153.613, Time=0.51 sec
 ARIMA(0,0,0)(0,1,0)[12]             : AIC=352.532, Time=0.02 sec
 ARIMA(1,0,1)(0,1,0)[12] intercept   : AIC=229.481, Time=0.19 sec
 ARIMA(1,0,1)(1,1,1)[12] intercept   : AIC=130.869, Time=0.87 sec
 ARIMA(1,0,1)(0,1,2)[12] intercept   : AIC=130.869, Time=2.64 sec
 ARIMA(1,0,1)(1,1,0)[12] intercept   : AIC=171.001, Time=0.58 sec
 ARIMA(1,0,1)(1,1,2)[12] intercept   : AIC=inf, Time=5.28 sec
 ARIMA(1,0,0)(0,1,1)[12] intercept   : AIC=128.161, Time=0.54 sec
 ARIMA(1,0,0)(0,1,0)[12] intercept   : AIC=231.910, Time=0.14 sec
 ARIMA(1,0,0)(1,1,1)[12] intercept   : AIC=130.088, Time=0.85 sec
 ARIMA(1,0,0)(0,1,2)[12] intercept   : AIC=130.077, Time=2.72 sec
 ARIMA(1,0,0)(1,1,2)[12] intercept   : AIC=inf, Time=4.53 sec
 ARIMA(0,0,0)(0,1,1)[12] intercept   : AIC=inf, Time=0.54 sec
 ARIMA(2,0,0)(0,1,1)[12] intercept   : AIC=129.256, Time=0.77 sec
 ARIMA(2,0,1)(0,1,1)[12] intercept   : AIC=130.368, Time=1.12 sec
 ARIMA(1,0,0)(0,1,1)[12]             : AIC=127.080, Time=0.34 sec
 ARIMA(1,0,0)(0,1,0)[12]             : AIC=229.921, Time=0.03 sec
 ARIMA(1,0,0)(1,1,1)[12]             : AIC=128.983, Time=0.48 sec
 ARIMA(1,0,0)(0,1,2)[12]             : AIC=128.969, Time=1.20 sec
 ARIMA(1,0,0)(1,1,0)[12]             : AIC=168.383, Time=0.14 sec
 ARIMA(1,0,0)(1,1,2)[12]             : AIC=inf, Time=3.90 sec
 ARIMA(0,0,0)(0,1,1)[12]             : AIC=253.182, Time=0.23 sec
 ARIMA(2,0,0)(0,1,1)[12]             : AIC=128.289, Time=0.54 sec
 ARIMA(1,0,1)(0,1,1)[12]             : AIC=127.942, Time=0.49 sec
 ARIMA(0,0,1)(0,1,1)[12]             : AIC=154.927, Time=0.37 sec
 ARIMA(2,0,1)(0,1,1)[12]             : AIC=129.418, Time=0.80 sec
```

```
Best model:  ARIMA(1,0,0)(0,1,1)[12]
Total fit time: 31.107 seconds
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                                     y   No. Observations:
260
Model:             SARIMAX(1, 0, 0)x(0, 1, [1], 12)   Log Likelihood
-60.540
Date:                              Sun, 14 Feb 2021   AIC
127.080
Time:                                     01:29:07   BIC
137.621
Sample:                                          0   HQIC
131.323
                                               - 260
Covariance Type:                                 opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.6364      0.045     14.258      0.000       0.549       0.724
ma.S.L12      -0.7568      0.050    -15.071      0.000      -0.855      -0.658
sigma2         0.0914      0.005     19.353      0.000       0.082       0.101
===================================================================================
===
Ljung-Box (Q):                       30.92   Jarque-Bera (JB):
446.58
Prob(Q):                              0.85   Prob(JB):
0.00
Heteroskedasticity (H):               0.76   Skew:
0.85
Prob(H) (two-sided):                  0.21   Kurtosis:
9.35
===================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```

1.0.4 Resíduos

[8]:
```
model.plot_diagnostics()
plt.show()
```



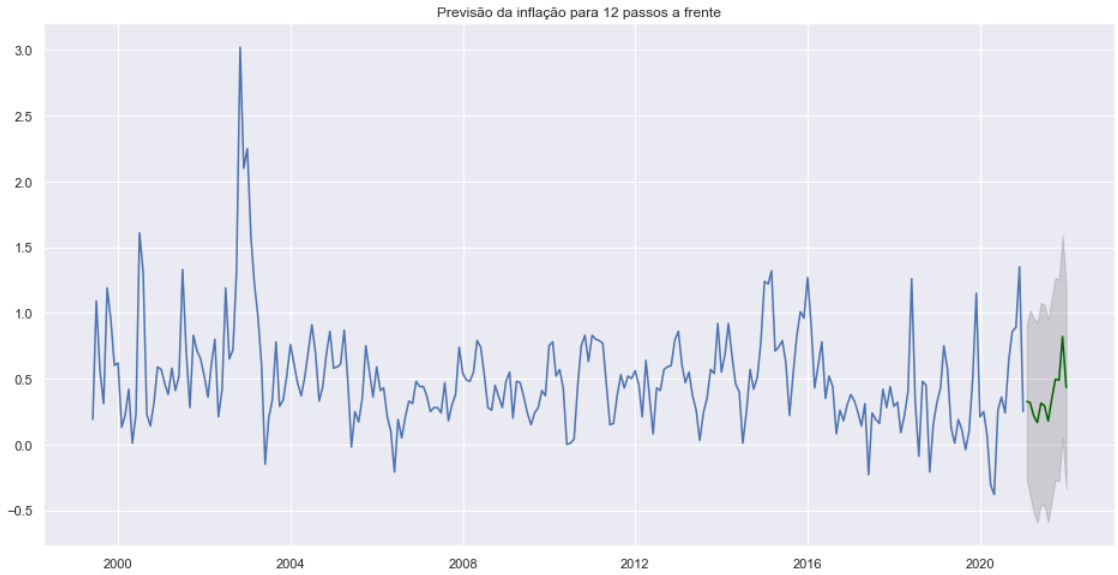
1.0.5 Análise

- Os erros residuais parecem flutuar em torno de uma média de zero e têm uma variância uniforme.
- O gráfico de densidade sugere distribuição normal com média zero.
- Todos os pontos devem ficar perfeitamente alinhados com a linha vermelha. Quaisquer desvios significativos implicariam que a distribuição é distorcida.
- O Correlograma, também conhecido como gráfico ACF, mostra que os erros residuais não são autocorrelacionados

1.0.6 Forecast

[9]:
```
# make series for plotting purpose
# Forecast
n_periods = 12
fc, confint = model.predict(n_periods=n_periods, return_conf_int=True)
index_of_fc = inflacao.index + relativedelta(months=+12)

teste = index_of_fc[-12:]

fc_series = pd.Series(fc, index=teste)
lower_series = pd.Series(confint[:, 0], index=teste)
upper_series = pd.Series(confint[:, 1], index=teste)

# Plot
plt.plot(inflacao.valor)
plt.plot(fc_series, color='darkgreen')
plt.fill_between(lower_series.index,
```

```
                lower_series,
                upper_series,
                color='k', alpha=.15)

plt.title("Previsão da inflação para 12 passos a frente")
plt.show()
```



[17]:
```
# valores da previsão
fc
```

[17]:
```
array([0.32697254, 0.31689144, 0.21717825, 0.16660201, 0.31397159,
       0.29425676, 0.17822873, 0.34855581, 0.49559291, 0.48871008,
       0.82034581, 0.43289792])
```