USDBRL

May 12, 2023

1 Machine Learning nas finanças

2 USD/BRL

3 1. Importação das bibliotecas

[4]:
```
"""
1°) Importação do pandas como pd para trabalhar com dados.
"""
import pandas as pd
"""
2°) Importação do numpy como np para trabalhar com matrizes e tudo mais.
"""
import numpy as np
"""
3°) Importação do matplotlib.pyplot como plt para fazer gráficos.
"""
import matplotlib.pyplot as plt
"""
4°) De matplotlib.ticker vamos importar o AutoMinorLocator e o MaxNLocator para␣
 ↪trabalhar com os "ticks"
    dos gráficos.
"""
import matplotlib.ticker as mticker
from matplotlib.ticker import AutoMinorLocator, MaxNLocator
"""
5°) De matplotlib.font_manager vamos importar FontProperties para criar fontes␣
 ↪de texto.
"""
from matplotlib.font_manager import FontProperties
"""
6°) Importação do seaborn para fazer gráficos
"""
import seaborn as sbn
"""
7°) Importação de pycaret.time_series para trabalhar com séries temporais
"""
```

```
from pycaret.time_series import *
from pycaret.internal.pycaret_experiment import TimeSeriesExperiment
from sktime.utils.plotting import plot_series
```

4 2. Trazendo dados para o python

[5]:
```
Dados = pd.read_csv("USD_BRL Dados Históricos (1).csv") # Ler os dados da␣
↪extensão .csv
Dados.drop(["Vol.", "Var%"], axis = 1, inplace = True) # Exclusão de duas␣
↪colunas desnecessárias
Dados.head(7) # Mostrar 7 linas
```

[5]:

	Data	Último	Abertura	Máxima	Mínima
0	11.05.2023	4,9308	4,9443	4,9839	4,9266
1	10.05.2023	4,9424	4,9882	4,9914	4,9345
2	09.05.2023	4,9862	5,0097	5,0374	4,9728
3	08.05.2023	5,0097	4,9531	5,0186	4,9410
4	05.05.2023	4,9518	4,9847	5,0080	4,9250
5	04.05.2023	4,9822	4,9959	5,0346	4,9694
6	03.05.2023	4,9949	5,0398	5,0524	4,9812

[6]:
```
Dados.head(20)
```

[6]:

	Data	Último	Abertura	Máxima	Mínima
0	11.05.2023	4,9308	4,9443	4,9839	4,9266
1	10.05.2023	4,9424	4,9882	4,9914	4,9345
2	09.05.2023	4,9862	5,0097	5,0374	4,9728
3	08.05.2023	5,0097	4,9531	5,0186	4,9410
4	05.05.2023	4,9518	4,9847	5,0080	4,9250
5	04.05.2023	4,9822	4,9959	5,0346	4,9694
6	03.05.2023	4,9949	5,0398	5,0524	4,9812
7	02.05.2023	5,0387	4,9893	5,0535	4,9889
8	01.05.2023	4,9889	4,9875	4,9895	4,9875
9	28.04.2023	4,9865	4,9776	5,0200	4,9776
10	27.04.2023	4,9771	5,0445	5,0490	4,9693
11	26.04.2023	5,0435	5,0542	5,0746	5,0353
12	25.04.2023	5,0497	5,0348	5,0845	5,0294
13	24.04.2023	5,0337	5,0481	5,0863	5,0339
14	21.04.2023	5,0486	5,0493	5,0493	5,0489
15	20.04.2023	5,0486	5,0759	5,0864	5,0360
16	19.04.2023	5,0753	4,9859	5,0895	4,9859
17	18.04.2023	4,9840	4,9415	4,9974	4,9097
18	17.04.2023	4,9413	4,9102	4,9573	4,8988
19	14.04.2023	4,9096	4,9279	4,9653	4,8922

[7]:
```
Dados.columns
```

[7]:
```
Index(['Data', 'Último', 'Abertura', 'Máxima', 'Mínima'], dtype='object')
```

Data: Data de cotagem

Último: Última avaliação do Dólar no dia

Abertura: Primeira avaliação do Dólar no dia

Máxima: Máxima avaliação do Dólar no dia

Mínima: Mínima avaliação do Dólar no dia

5 3. Pré-Processamento de dados

5.1 3.1 Dtypes

[8]:
```
Dados.dtypes
```

[8]:
```
Data        object
Último      object
Abertura    object
Máxima      object
Mínima      object
dtype: object
```

[9]:
```
Dados = Dados.replace(",",".", regex = True) # Tudo que é vírgula vira ponto
Dados["Data"] = pd.to_datetime(Dados["Data"], format = "%d.%m.%Y")#␣
 ↪Tranformando no formato de data
Dados['Data'] = Dados['Data'].dt.strftime('%Y-%m-%d')
Dados["Último"] = Dados["Último"].astype(float) # Tranformando em float
Dados["Abertura"] = Dados["Abertura"].astype(float)
Dados["Máxima"] = Dados["Máxima"].astype(float)
Dados["Mínima"] = Dados["Mínima"].astype(float)
Dados.dtypes
```

[9]:
```
Data         object
Último      float64
Abertura    float64
Máxima      float64
Mínima      float64
dtype: object
```

5.2 3.2 Valores nulos

[10]:
```
Valores_nulos_percentual = 100*(Dados.isnull().sum()/len(Dados["Mínima"]))
print(Valores_nulos_percentual)
```

```
Data        0.0
Último      0.0
Abertura    0.0
```

```
Máxima      0.0
Mínima      0.0
dtype: float64
```

Não há nenhum valor nulo no dataset!

5.3 3.3 Valor médio do Dólar no dia

[11]:
```
Dados["Média"] = Dados[["Máxima", "Mínima"]].mean(axis = 1) # Tirando uma média entre duas colunas
Dados.head(5)
```

[11]:

	Data	Último	Abertura	Máxima	Mínima	Média
0	2023-05-11	4.9308	4.9443	4.9839	4.9266	4.95525
1	2023-05-10	4.9424	4.9882	4.9914	4.9345	4.96295
2	2023-05-09	4.9862	5.0097	5.0374	4.9728	5.00510
3	2023-05-08	5.0097	4.9531	5.0186	4.9410	4.97980
4	2023-05-05	4.9518	4.9847	5.0080	4.9250	4.96650

5.4 3.4 Análise de dados

[ ]:
```
```

[23]:
```
datatoexcel = pd.ExcelWriter('Dados.xlsx')
Dados.to_excel(datatoexcel)
datatoexcel.save()
print('DataFrame is written to Excel File successfully.')
```

```
DataFrame is written to Excel File successfully.
```

5.5 4. Previsão da série temporal de câmbio

[13]:
```
Serie_temporal = Dados[["Data", "Média"]]
Serie_temporal.index = pd.date_range(end = "2023-05-11", periods=1830, freq ="D")
Serie_temporal = Serie_temporal.drop("Data", axis = 1)
```

[14]:
```
Serie_temporal = Serie_temporal[::-1]
```

[15]:
```
Media_correta = []
for i in range(1830):
    Media_correta.append(Serie_temporal["Média"][i])
Serie_temporal = Serie_temporal[::-1]
```

[16]:
```
Serie_temporal["Media_correta"] = Media_correta
Serie_temporal.drop(["Média"], axis = 1, inplace = True)
Serie_temporal
```

[16]:
```
            Media_correta
2018-05-08        3.52915
2018-05-09        3.53435
2018-05-10        3.58420
2018-05-11        3.49005
2018-05-12        3.46195
…                   …
2023-05-07        4.96650
2023-05-08        4.97980
2023-05-09        5.00510
2023-05-10        4.96295
2023-05-11        4.95525

[1830 rows x 1 columns]
```

[17]:
```
setup(Serie_temporal, fh=120, fold=13, seasonal_period="D", n_jobs = -1,␣
↪use_gpu = True); # Criando um setup
```

```
<pandas.io.formats.style.Styler at 0x2c63f1d7fa0>
```

[18]:
```
#Compare = compare_models(exclude=['auto_arima']) # Comparar modelos
```

[19]:
```
theta = create_model("theta") # Criar o melhor modelo
```

```
<pandas.io.formats.style.Styler at 0x2c63f1d79a0>
```

[20]:
```
final = finalize_model(theta) # finalizar o modelo
```

[21]:
```
"""
Predições
"""
pred = predict_model(final, fh = 60)
pred = pd.DataFrame(pred, columns = ["Data", "Media_correta"]) # Transformando␣
↪em DataFrame
pred["Data"] = pred.index.to_timestamp()
pred
```

[21]:
```
                 Data  Media_correta
2023-05-12 2023-05-12         4.9577
2023-05-13 2023-05-13         4.9623
2023-05-14 2023-05-14         4.9606
2023-05-15 2023-05-15         4.9624
2023-05-16 2023-05-16         4.9601
2023-05-17 2023-05-17         4.9611
2023-05-18 2023-05-18         4.9614
2023-05-19 2023-05-19         4.9630
2023-05-20 2023-05-20         4.9677
2023-05-21 2023-05-21         4.9660
```

2023-05-22	2023-05-22	4.9678
2023-05-23	2023-05-23	4.9654
2023-05-24	2023-05-24	4.9664
2023-05-25	2023-05-25	4.9667
2023-05-26	2023-05-26	4.9684
2023-05-27	2023-05-27	4.9731
2023-05-28	2023-05-28	4.9713
2023-05-29	2023-05-29	4.9731
2023-05-30	2023-05-30	4.9708
2023-05-31	2023-05-31	4.9718
2023-06-01	2023-06-01	4.9721
2023-06-02	2023-06-02	4.9738
2023-06-03	2023-06-03	4.9784
2023-06-04	2023-06-04	4.9767
2023-06-05	2023-06-05	4.9785
2023-06-06	2023-06-06	4.9762
2023-06-07	2023-06-07	4.9772
2023-06-08	2023-06-08	4.9775
2023-06-09	2023-06-09	4.9791
2023-06-10	2023-06-10	4.9838
2023-06-11	2023-06-11	4.9821
2023-06-12	2023-06-12	4.9839
2023-06-13	2023-06-13	4.9815
2023-06-14	2023-06-14	4.9825
2023-06-15	2023-06-15	4.9828
2023-06-16	2023-06-16	4.9845
2023-06-17	2023-06-17	4.9892
2023-06-18	2023-06-18	4.9875
2023-06-19	2023-06-19	4.9892
2023-06-20	2023-06-20	4.9869
2023-06-21	2023-06-21	4.9879
2023-06-22	2023-06-22	4.9882
2023-06-23	2023-06-23	4.9899
2023-06-24	2023-06-24	4.9945
2023-06-25	2023-06-25	4.9928
2023-06-26	2023-06-26	4.9946
2023-06-27	2023-06-27	4.9923
2023-06-28	2023-06-28	4.9933
2023-06-29	2023-06-29	4.9936
2023-06-30	2023-06-30	4.9952
2023-07-01	2023-07-01	4.9999
2023-07-02	2023-07-02	4.9982
2023-07-03	2023-07-03	5.0000
2023-07-04	2023-07-04	4.9976
2023-07-05	2023-07-05	4.9986
2023-07-06	2023-07-06	4.9989
2023-07-07	2023-07-07	5.0006

```
2023-07-08 2023-07-08         5.0053
2023-07-09 2023-07-09         5.0036
2023-07-10 2023-07-10         5.0053
```

[22]:
```
"""
Criação da primeira fonte de texto para colocar como fonte dos labels
"""
font1 = {"family": "serif", "weight": "bold", "color": "gray", "size": 14}
"""
Criação da segunda fonte de texto para colocar como fonte da legenda
"""
font2 = FontProperties(family = "serif",
                       weight = "bold",
                       size = 14)
"""
Cria um "lugar" com size (9, 7) para alocar a figura
"""
fig, axs = plt.subplots(figsize = (14, 7))
"Plot do gráfico"
axs.plot(pred["Data"],
         pred["Media_correta"],
         color = "blue",
         linewidth = 1.5,
         label = "Previsão (2023-05-12 até 2023-07-10)")
axs.grid(False)
"""
Definindo a "grossura" e a cor do eixos
"""
for axis in ["left", "right", "top", "bottom"]:
    axs.spines[axis].set_linewidth(2)
    axs.spines[axis].set_color("gray")
"""
Trabalha com os ticks do gráfico
"""
axs.xaxis.set_minor_locator(AutoMinorLocator())
axs.yaxis.set_minor_locator(AutoMinorLocator())
axs.tick_params(axis = "both", direction = "in", labelcolor = "gray", labelsize␣
 ↪= 14, left = True, bottom = True, top = True, right = True)
axs.tick_params(which = "major", direction = "in", color = "gray", length = 5.
 ↪4, width = 2.5, left = True, bottom = False, top = False, right = True)
axs.tick_params(which = "minor", direction = "in", color = "gray", length=4,␣
 ↪width = 2, left = True, bottom = True, top = True, right = True)
"""
Descrição para cada eixo
"""
axs.set_xlabel("Data", fontdict = font1)
axs.set_ylabel("USD/BRL", fontdict = font1)
```

```
"""
plt.rcParams["axes.labelweight"] = "bold" mostra em negrito os números nos␣
↪eixos.
"""
plt.rcParams["axes.labelweight"] = "bold"
plt.legend(frameon = False, prop = font2, labelcolor = "gray")
"""
Definindo um fundo branco para a imagem
"""
fig.patch.set_facecolor("white")
Cor_fundo = plt.gca()
Cor_fundo.set_facecolor("white")
Cor_fundo.patch.set_alpha(1)
"""
Mostrar o gráfico
"""
plt.show()
```