ibovespa_

May 12, 2023

1 Machine Learning nas finanças

2 IBOVESPA

3 1. Importação das bibliotecas

[52]:

```
"""
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

[53]:
```
Dados = pd.read_csv("Dados Históricos - Ibovespa.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 linhas
```

[53]:
```
         Data   Último  Abertura   Máxima   Mínima
0  12.05.2023  108.464   108.256  108.817  107.497
1  11.05.2023  108.256   107.446  108.667  106.419
2  10.05.2023  107.448   107.114  107.744  106.538
3  09.05.2023  107.114   106.028  107.731  105.549
4  08.05.2023  106.042   105.161  106.716  105.161
5  05.05.2023  105.148   102.175  105.306  102.175
6  04.05.2023  102.174   101.798  103.321  101.063
```

[54]:
```
Dados.head(20)
```

[54]:
```
          Data   Último  Abertura   Máxima   Mínima
0   12.05.2023  108.464   108.256  108.817  107.497
1   11.05.2023  108.256   107.446  108.667  106.419
2   10.05.2023  107.448   107.114  107.744  106.538
3   09.05.2023  107.114   106.028  107.731  105.549
4   08.05.2023  106.042   105.161  106.716  105.161
5   05.05.2023  105.148   102.175  105.306  102.175
6   04.05.2023  102.174   101.798  103.321  101.063
7   03.05.2023  101.797   101.927  102.331  101.433
8   02.05.2023  101.927   104.431  104.447  101.569
9   28.04.2023  104.432   102.923  104.432  102.449
10  27.04.2023  102.923   102.310  103.177  101.975
11  26.04.2023  102.312   103.220  103.668  102.233
12  25.04.2023  103.220   103.947  103.947  102.633
13  24.04.2023  103.947   104.367  104.822  103.247
14  20.04.2023  104.367   103.913  104.615  103.087
15  19.04.2023  103.913   106.149  106.149  103.604
16  18.04.2023  106.163   106.023  106.475  105.122
17  17.04.2023  106.016   106.279  106.830  105.623
18  14.04.2023  106.279   106.458  106.701  104.934
19  13.04.2023  106.458   106.890  107.037  106.220
```

[55]:
```
Dados.columns
```

```
[55]: 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

```
[56]: Dados.dtypes
```

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

```
[57]: 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
```

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

5.2 3.2 Valores nulos

```
[58]: 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

[59]:
```
Dados["Média"] = Dados[["Máxima", "Mínima"]].mean(axis = 1) # Tirando uma média␣
 ↪entre duas colunas
Dados.head(5)
```

[59]:
```
         Data   Último  Abertura   Máxima   Mínima     Média
0  2023-05-12  108.464   108.256  108.817  107.497  108.1570
1  2023-05-11  108.256   107.446  108.667  106.419  107.5430
2  2023-05-10  107.448   107.114  107.744  106.538  107.1410
3  2023-05-09  107.114   106.028  107.731  105.549  106.6400
4  2023-05-08  106.042   105.161  106.716  105.161  105.9385
```

5.4 3.4 Análise de dados

[60]:
```
Dados.shape
```

[60]:
```
(1741, 6)
```

[61]:
```
datatoexcel = pd.ExcelWriter('IBOVESPA.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

[62]:
```
Serie_temporal = Dados[["Data", "Média"]]
Serie_temporal.index = pd.date_range(end = "2023-05-11", periods=1741, freq =␣
 ↪"D")
Serie_temporal = Serie_temporal.drop("Data", axis = 1)
```

[63]:
```
Serie_temporal = Serie_temporal[::-1]
```

[64]:
```
Media_correta = []
for i in range(1741):
    Media_correta.append(Serie_temporal["Média"][i])
Serie_temporal = Serie_temporal[::-1]
```

[65]:
```
Serie_temporal["Media_correta"] = Media_correta
Serie_temporal.drop(["Média"], axis = 1, inplace = True)
Serie_temporal
```

[65]:

```
            Media_correta
2018-08-05        52.1910
2018-08-06        51.6815
2018-08-07        50.8125
2018-08-08        52.0325
2018-08-09        53.3560
…                    …
2023-05-07       105.9385
2023-05-08       106.6400
2023-05-09       107.1410
2023-05-10       107.5430
2023-05-11       108.1570

[1741 rows x 1 columns]
```

[66]:

```
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 0x1df130b5d30>
```

```
INFO:logs:self.master_model_container: 0
INFO:logs:self.display_container: 1
INFO:logs:Pipeline(memory=None,
         steps=[('dtypes',
                 DataTypes_Auto_infer(categorical_features=[],
                                      display_types=False, features_todrop=[],
                                      float_dtype='float64', id_columns=[],
                                      ml_usecase='regression',
                                      numerical_features=[],
                                      target='Media_correta',
                                      time_features=[])),
                ('imputer',
                 Simple_Imputer(categorical_strategy='most frequent',
                                fill_value_categorical='not_available',
                                fil…
                ('scaling', 'passthrough'), ('P_transform', 'passthrough'),
                ('binn', 'passthrough'), ('rem_outliers', 'passthrough'),
                ('cluster_all', 'passthrough'),
                ('dummy', Dummify(target='Media_correta')),
                ('fix_perfect', 'passthrough'),
                ('clean_names', Clean_Colum_Names()),
                ('feature_select', 'passthrough'), ('fix_multi', 'passthrough'),
                ('dfs', 'passthrough'), ('pca', 'passthrough')],
         verbose=False)
INFO:logs:setup() successfully completed…
```

[67]:

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

```
<pandas.io.formats.style.Styler at 0x1df12f37df0>
```

```
INFO:logs:master_model_container: 29
INFO:logs:display_container: 2
INFO:logs:BaseCdsDtForecaster(degree=1, deseasonal_model='additive',
                    regressor=OrthogonalMatchingPursuit(fit_intercept=True,
                                                        n_nonzero_coefs=None,
                                                        normalize='deprecated',
                                                        precompute='auto',
                                                        tol=None),
                    sp=7, window_length=7)
INFO:logs:compare_models() successfully
completed…
```

[68]:
```
omp_cds_dt = create_model("omp_cds_dt") # Criar o melhor modelo
```

```
<pandas.io.formats.style.Styler at 0x1df12d94f40>
```

```
INFO:logs:master_model_container: 30
INFO:logs:display_container: 3
INFO:logs:BaseCdsDtForecaster(degree=1, deseasonal_model='additive',
                    regressor=OrthogonalMatchingPursuit(fit_intercept=True,
                                                        n_nonzero_coefs=None,
                                                        normalize='deprecated',
                                                        precompute='auto',
                                                        tol=None),
                    sp=7, window_length=7)
INFO:logs:create_model() successfully
completed…
```

[69]:
```
final = finalize_model(omp_cds_dt) # finalizar o modelo
```

```
INFO:logs:Initializing finalize_model()
INFO:logs:finalize_model(self=<pycaret.internal.pycaret_experiment.time_series_e
xperiment.TSForecastingExperiment object at 0x000001DF0CF4FAC0>,
estimator=BaseCdsDtForecaster(degree=1, deseasonal_model='additive',
                    regressor=OrthogonalMatchingPursuit(fit_intercept=True,
                                                        n_nonzero_coefs=None,
                                                        normalize='deprecated',
                                                        precompute='auto',
                                                        tol=None),
                    sp=7, window_length=7), fit_kwargs=None, groups=None,
model_only=True, display=None)
INFO:logs:Finalizing BaseCdsDtForecaster(degree=1, deseasonal_model='additive',
                    regressor=OrthogonalMatchingPursuit(fit_intercept=True,
                                                        n_nonzero_coefs=None,
                                                        normalize='deprecated',
                                                        precompute='auto',
                                                        tol=None),
                    sp=7, window_length=7)
INFO:logs:Initializing create_model()
```

```
INFO:logs:create_model(self=<pycaret.internal.pycaret_experiment.time_series_exp
eriment.TSForecastingExperiment object at 0x000001DF0CF4FAC0>,
estimator=BaseCdsDtForecaster(degree=1, deseasonal_model='additive',
                    regressor=OrthogonalMatchingPursuit(fit_intercept=True,
                                                        n_nonzero_coefs=None,
                                                        normalize='deprecated',
                                                        precompute='auto',
                                                        tol=None),
                    sp=7, window_length=7), fold=None, round=4,
cross_validation=True, predict=True, fit_kwargs={}, groups=None, refit=True,
probability_threshold=None, verbose=False, system=False,
add_to_model_list=False, metrics=None, display=None, kwargs={})
INFO:logs:Checking exceptions
INFO:logs:Importing libraries
INFO:logs:Copying training dataset
INFO:logs:Defining folds
INFO:logs:Declaring metric variables
INFO:logs:Importing untrained model
INFO:logs:Declaring custom model
INFO:logs:OrthogonalMatchingPursuit Imported successfully
INFO:logs:Starting cross validation
INFO:logs:Cross validating with ExpandingWindowSplitter(fh=array([  1,   2, …,
119, 120]),
            initial_window=None, step_length=120), n_jobs=1
INFO:logs:Calculating mean and std
INFO:logs:Creating metrics dataframe
INFO:logs:Finalizing model
INFO:logs:Uploading results into container
INFO:logs:master_model_container: 30
INFO:logs:display_container: 4
INFO:logs:BaseCdsDtForecaster(degree=1, deseasonal_model='additive',
                    regressor=OrthogonalMatchingPursuit(fit_intercept=True,
                                                        n_nonzero_coefs=None,
                                                        normalize='deprecated',
                                                        precompute='auto',
                                                        tol=None),
                    sp=7, window_length=7)
INFO:logs:create_model() successfully
completed…
INFO:logs:master_model_container: 30
INFO:logs:display_container: 3
INFO:logs:BaseCdsDtForecaster(degree=1, deseasonal_model='additive',
                    regressor=OrthogonalMatchingPursuit(fit_intercept=True,
                                                        n_nonzero_coefs=None,
                                                        normalize='deprecated',
                                                        precompute='auto',
                                                        tol=None),
                    sp=7, window_length=7)
```

```
INFO:logs:finalize_model() succesfully
completed…
```

[70]:

```
"""
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
```

[70]:

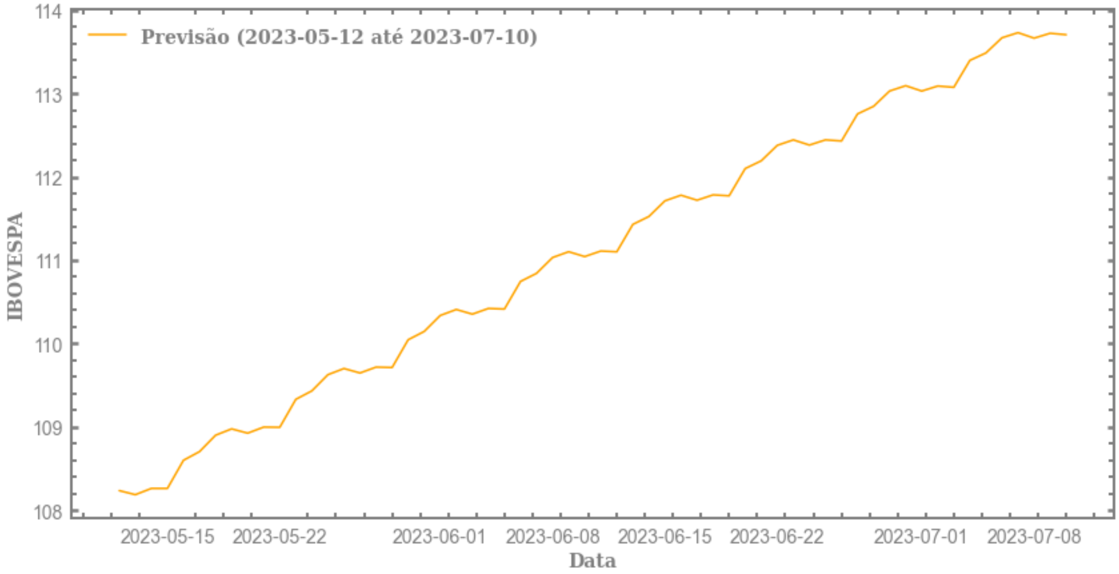
```
                  Data  Media_correta
2023-05-12 2023-05-12       108.2355
2023-05-13 2023-05-13       108.1866
2023-05-14 2023-05-14       108.2617
2023-05-15 2023-05-15       108.2617
2023-05-16 2023-05-16       108.5993
2023-05-17 2023-05-17       108.7033
2023-05-18 2023-05-18       108.9000
2023-05-19 2023-05-19       108.9761
2023-05-20 2023-05-20       108.9248
2023-05-21 2023-05-21       108.9975
2023-05-22 2023-05-22       108.9951
2023-05-23 2023-05-23       109.3305
2023-05-24 2023-05-24       109.4321
2023-05-25 2023-05-25       109.6264
2023-05-26 2023-05-26       109.7002
2023-05-27 2023-05-27       109.6467
2023-05-28 2023-05-28       109.7171
2023-05-29 2023-05-29       109.7124
2023-05-30 2023-05-30       110.0455
2023-05-31 2023-05-31       110.1449
2023-06-01 2023-06-01       110.3369
2023-06-02 2023-06-02       110.4085
2023-06-03 2023-06-03       110.3527
2023-06-04 2023-06-04       110.4210
2023-06-05 2023-06-05       110.4141
2023-06-06 2023-06-06       110.7449
2023-06-07 2023-06-07       110.8422
2023-06-08 2023-06-08       111.0320
2023-06-09 2023-06-09       111.1015
2023-06-10 2023-06-10       111.0435
2023-06-11 2023-06-11       111.1096
2023-06-12 2023-06-12       111.1006
2023-06-13 2023-06-13       111.4293
2023-06-14 2023-06-14       111.5245
```

```
2023-06-15 2023-06-15       111.7122
2023-06-16 2023-06-16       111.7796
2023-06-17 2023-06-17       111.7195
2023-06-18 2023-06-18       111.7836
2023-06-19 2023-06-19       111.7725
2023-06-20 2023-06-20       112.0992
2023-06-21 2023-06-21       112.1923
2023-06-22 2023-06-22       112.3780
2023-06-23 2023-06-23       112.4434
2023-06-24 2023-06-24       112.3813
2023-06-25 2023-06-25       112.4433
2023-06-26 2023-06-26       112.4303
2023-06-27 2023-06-27       112.7550
2023-06-28 2023-06-28       112.8461
2023-06-29 2023-06-29       113.0299
2023-06-30 2023-06-30       113.0933
2023-07-01 2023-07-01       113.0293
2023-07-02 2023-07-02       113.0894
2023-07-03 2023-07-03       113.0744
2023-07-04 2023-07-04       113.3972
2023-07-05 2023-07-05       113.4864
2023-07-06 2023-07-06       113.6683
2023-07-07 2023-07-07       113.7298
2023-07-08 2023-07-08       113.6639
2023-07-09 2023-07-09       113.7222
2023-07-10 2023-07-10       113.7053
```

[71]:

```python
"""
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 = "orange",
         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("IBOVESPA", 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()
```


Previsão (2023-05-12 até 2023-07-10)
IBOVESPA
114
113
112
111
110
109
108
2023-05-15
2023-05-22
2023-06-01
2023-06-08
2023-06-15
2023-06-22
2023-07-01
2023-07-08
Data