In [218…

```
# 1 - BIBLIOTECAS e PACOTES

import numpy as np
import seaborn as sns
import pandas as pd
from pandas import DataFrame
from pandas.util.testing import assert_frame_equal
from pandas_datareader import data
import matplotlib.pyplot as plt
import datetime
import math
```

In [219…

```
from scipy import stats
```

In [220…

```
import plotly.express as px
import plotly.figure_factory as ff
from copy import copy
```

In [221…

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [222…

```
import sklearn.metrics
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

In [223…

```
from tensorflow import keras
```

In [224…

```
# 2 - PROCESSAMENTO/TRATAMENTO DOS DADOS
```

In [225…

```
milho_df = pd.read_excel('Milho-CEPEA-ESALQ.xlsx')
milho_df
```

Out[225…

	INDICADOR DO MILHO ESALQ/BM&FBOVESPA	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unna
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	Fonte: Cepea	NaN	NaN	NaN	NaN	NaN	NaN	
2	Data	À vista R$	À vista US$	NaN	NaN	NaN	NaN	
3	02/08/2004	18.24	5.98	NaN	NaN	NaN	NaN	
4	03/08/2004	18.04	5.91	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	
4125	25/02/2021	85.59	15.55	NaN	NaN	NaN	NaN	
4126	26/02/2021	85.41	15.3	NaN	NaN	NaN	NaN	
4127	01/03/2021	85.59	15.29	NaN	NaN	NaN	NaN	
4128	02/03/2021	86.11	15.2	NaN	NaN	NaN	NaN	
4129	03/03/2021	87.06	15.14	NaN	NaN	NaN	NaN	

4130 rows × 10 columns

In [226…

```
milho_df = milho_df.drop(milho_df.index[0:3])
milho_df = milho_df.drop(columns=milho_df.columns[3:])
milho_df
```

Out[226…

	INDICADOR DO MILHO ESALQ/BM&FBOVESPA	Unnamed: 1	Unnamed: 2
3	02/08/2004	18.24	5.98
4	03/08/2004	18.04	5.91
5	04/08/2004	18.02	5.9
6	05/08/2004	18.06	5.89
7	06/08/2004	18.13	5.98
...	...	...	...
4125	25/02/2021	85.59	15.55
4126	26/02/2021	85.41	15.3
4127	01/03/2021	85.59	15.29
4128	02/03/2021	86.11	15.2
4129	03/03/2021	87.06	15.14

4127 rows × 3 columns

In [227…

```
milho_df.rename(columns= {'INDICADOR DO MILHO ESALQ/BM&FBOVESPA': 'Data'}, inplace=True
milho_df.rename(columns= {'Unnamed: 1': 'milho_reais'}, inplace=True)
milho_df.rename(columns= {'Unnamed: 2': 'milho_dolares'}, inplace=True)
milho_df
```

Out[227…

	Data	milho_reais	milho_dolares
3	02/08/2004	18.24	5.98
4	03/08/2004	18.04	5.91
5	04/08/2004	18.02	5.9
6	05/08/2004	18.06	5.89
7	06/08/2004	18.13	5.98
...	...	...	...
4125	25/02/2021	85.59	15.55
4126	26/02/2021	85.41	15.3
4127	01/03/2021	85.59	15.29
4128	02/03/2021	86.11	15.2
4129	03/03/2021	87.06	15.14

4127 rows × 3 columns

In [228…

```
milho_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4127 entries, 3 to 4129
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Data           4127 non-null   object
 1   milho_reais    4127 non-null   object
 2   milho_dolares  4127 non-null   object
dtypes: object(3)
memory usage: 129.0+ KB
```

In [229…

```
milho_df['milho_reais'] = milho_df['milho_reais'].astype(float)
milho_df['milho_dolares'] = milho_df['milho_dolares'].astype(float)
milho_df
```

Out[229…

	Data	milho_reais	milho_dolares
3	02/08/2004	18.24	5.98
4	03/08/2004	18.04	5.91
5	04/08/2004	18.02	5.90
6	05/08/2004	18.06	5.89
7	06/08/2004	18.13	5.98
...	...	...	...
4125	25/02/2021	85.59	15.55
4126	26/02/2021	85.41	15.30
4127	01/03/2021	85.59	15.29
4128	02/03/2021	86.11	15.20
4129	03/03/2021	87.06	15.14

4127 rows × 3 columns

In [230…

```
milho_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4127 entries, 3 to 4129
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Data           4127 non-null   object 
 1   milho_reais    4127 non-null   float64
 2   milho_dolares  4127 non-null   float64
dtypes: float64(2), object(1)
memory usage: 129.0+ KB
```

In [231…

```
milho_df['Data']
```

Out[231…

```
3       02/08/2004
4       03/08/2004
5       04/08/2004
6       05/08/2004
7       06/08/2004
           ...
```

```
4125    25/02/2021
4126    26/02/2021
4127    01/03/2021
4128    02/03/2021
4129    03/03/2021
Name: Data, Length: 4127, dtype: object
```

In [232…

```
milho_df['Data'] = pd.to_datetime(milho_df['Data'],dayfirst=True)
milho_df = milho_df.sort_values(by = ['Data'])
milho_df['Data']
```

Out[232…

```
3      2004-08-02
4      2004-08-03
5      2004-08-04
6      2004-08-05
7      2004-08-06
          ...
4125   2021-02-25
4126   2021-02-26
4127   2021-03-01
4128   2021-03-02
4129   2021-03-03
Name: Data, Length: 4127, dtype: datetime64[ns]
```

In [233…

```
milho_df.index
```

Out[233…

```
Int64Index([   3,    4,    5,    6,    7,    8,    9,   10,   11,   12,
            ...
            4120, 4121, 4122, 4123, 4124, 4125, 4126, 4127, 4128, 4129],
           dtype='int64', length=4127)
```

In [234…

```
milho_df.index = pd.to_datetime(milho_df.Data)
milho_df.index.to_period('D')
milho_df.index
```

Out[234…

```
DatetimeIndex(['2004-08-02', '2004-08-03', '2004-08-04', '2004-08-05',
               '2004-08-06', '2004-08-09', '2004-08-10', '2004-08-11',
               '2004-08-12', '2004-08-13',
               ...
               '2021-02-18', '2021-02-19', '2021-02-22', '2021-02-23',
               '2021-02-24', '2021-02-25', '2021-02-26', '2021-03-01',
               '2021-03-02', '2021-03-03'],
              dtype='datetime64[ns]', name='Data', length=4127, freq=None)
```

In [235…

```
milho_df.isnull().sum()
```

Out[235…

```
Data            0
milho_reais     0
milho_dolares   0
dtype: int64
```

In [236…

```
ccmfut_df = pd.read_excel('CCMFUT-ProfitChart.xlsx')
ccmfut_df
```

Out[236…

	Data	Abertura	Máxima	Mínima	Fechamento	Volume Financeiro
0	2021-03-05	94.10	96.46	94.10	95.85	380377381.5
1	2021-03-04	91.00	94.72	90.40	94.20	325967202.0
2	2021-03-03	89.00	91.10	88.86	91.05	211768164.0
3	2021-03-02	88.80	89.06	88.54	88.94	95795617.5

	Data	Abertura	Máxima	Mínima	Fechamento	Volume Financeiro
4	2021-03-01	88.92	89.18	88.00	88.70	133054398.0
...	...	...	...	...	...	...
2914	2008-10-02	22.64	22.64	22.64	22.64	11295.0
2915	2008-09-30	22.55	22.55	22.55	22.55	112500.0
2916	2008-09-26	22.68	22.68	22.68	22.68	565875.0
2917	2008-09-22	22.64	22.64	22.64	22.64	112950.0
2918	2008-09-19	22.64	22.64	22.64	22.64	1129500.0

2919 rows × 6 columns

In [237…

```
ccmfut_df.rename(columns= {'Abertura': 'ccmfut_abertura'}, inplace=True)
ccmfut_df.rename(columns= {'Máxima': 'ccmfut_máxima'}, inplace=True)
ccmfut_df.rename(columns= {'Mínima': 'ccmfut_mínima'}, inplace=True)
ccmfut_df.rename(columns= {'Fechamento': 'ccmfut_fechamento'}, inplace=True)
ccmfut_df.rename(columns= {'Volume Financeiro': 'ccmfut_volume_fin'}, inplace=True)
ccmfut_df
```

Out[237…

	Data	ccmfut_abertura	ccmfut_máxima	ccmfut_mínima	ccmfut_fechamento	ccmfut_volume_fin
0	2021-03-05	94.10	96.46	94.10	95.85	380377381.5
1	2021-03-04	91.00	94.72	90.40	94.20	325967202.0
2	2021-03-03	89.00	91.10	88.86	91.05	211768164.0
3	2021-03-02	88.80	89.06	88.54	88.94	95795617.5
4	2021-03-01	88.92	89.18	88.00	88.70	133054398.0
...	...	...	...	...	...	...
2914	2008-10-02	22.64	22.64	22.64	22.64	11295.0
2915	2008-09-30	22.55	22.55	22.55	22.55	112500.0
2916	2008-09-26	22.68	22.68	22.68	22.68	565875.0
2917	2008-09-22	22.64	22.64	22.64	22.64	112950.0
2918	2008-09-19	22.64	22.64	22.64	22.64	1129500.0

2919 rows × 6 columns

In [238…

```
ccmfut_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2919 entries, 0 to 2918
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Data                2919 non-null   datetime64[ns]
 1   ccmfut_abertura     2919 non-null   float64
 2   ccmfut_máxima       2919 non-null   float64
 3   ccmfut_mínima       2919 non-null   float64
 4   ccmfut_fechamento   2919 non-null   float64
 5   ccmfut_volume_fin   2919 non-null   float64
dtypes: datetime64[ns](1), float64(5)
memory usage: 137.0 KB
```

In [239…

```
ccmfut_df = ccmfut_df.drop(ccmfut_df.index[0:2])
ccmfut_df
```

Out[239…

	Data	ccmfut_abertura	ccmfut_máxima	ccmfut_mínima	ccmfut_fechamento	ccmfut_volume_fin
2	2021-03-03	89.00	91.10	88.86	91.05	211768164.0
3	2021-03-02	88.80	89.06	88.54	88.94	95795617.5
4	2021-03-01	88.92	89.18	88.00	88.70	133054398.0
5	2021-02-26	89.61	89.64	88.86	88.86	69200707.5
6	2021-02-25	89.45	89.72	88.81	89.63	106361550.0
...	...	...	...	...	...	...
2914	2008-10-02	22.64	22.64	22.64	22.64	11295.0
2915	2008-09-30	22.55	22.55	22.55	22.55	112500.0
2916	2008-09-26	22.68	22.68	22.68	22.68	565875.0
2917	2008-09-22	22.64	22.64	22.64	22.64	112950.0
2918	2008-09-19	22.64	22.64	22.64	22.64	1129500.0

2917 rows × 6 columns

In [240…

```
ccmfut_df = ccmfut_df.sort_values(by = ['Data'])
ccmfut_df
```

Out[240…

	Data	ccmfut_abertura	ccmfut_máxima	ccmfut_mínima	ccmfut_fechamento	ccmfut_volume_fin
2918	2008-09-19	22.64	22.64	22.64	22.64	1129500.0
2917	2008-09-22	22.64	22.64	22.64	22.64	112950.0

	Data	ccmfut_abertura	ccmfut_máxima	ccmfut_mínima	ccmfut_fechamento	ccmfut_volume_fin
2916	2008-09-26	22.68	22.68	22.68	22.68	565875.0
2915	2008-09-30	22.55	22.55	22.55	22.55	112500.0
2914	2008-10-02	22.64	22.64	22.64	22.64	11295.0
...	...	...	...	...	...	...
6	2021-02-25	89.45	89.72	88.81	89.63	106361550.0
5	2021-02-26	89.61	89.64	88.86	88.86	69200707.5
4	2021-03-01	88.92	89.18	88.00	88.70	133054398.0
3	2021-03-02	88.80	89.06	88.54	88.94	95795617.5
2	2021-03-03	89.00	91.10	88.86	91.05	211768164.0

2917 rows × 6 columns

In [241…
```
ccmfut_df.index
```

Out[241…
```
Int64Index([2918, 2917, 2916, 2915, 2914, 2913, 2912, 2911, 2910, 2909,
            ...
              11,   10,    9,    8,    7,    6,    5,    4,    3,    2],
           dtype='int64', length=2917)
```

In [242…
```
ccmfut_df.index = pd.to_datetime(ccmfut_df.Data)
ccmfut_df.index.to_period('D')
ccmfut_df.index
```

Out[242…
```
DatetimeIndex(['2008-09-19', '2008-09-22', '2008-09-26', '2008-09-30',
               '2008-10-02', '2008-10-03', '2008-10-06', '2008-10-07',
               '2008-10-08', '2008-10-09',
               ...
               '2021-02-18', '2021-02-19', '2021-02-22', '2021-02-23',
               '2021-02-24', '2021-02-25', '2021-02-26', '2021-03-01',
               '2021-03-02', '2021-03-03'],
              dtype='datetime64[ns]', name='Data', length=2917, freq=None)
```

In [243…
```
ccmfut_df.isnull().sum()
```

Out[243…
```
Data                 0
ccmfut_abertura      0
ccmfut_máxima        0
ccmfut_mínima        0
ccmfut_fechamento    0
ccmfut_volume_fin    0
dtype: int64
```

In [244…
```
milho_df.rename(columns= {'Data': 'data'}, inplace=True)
ccmfut_df.rename(columns= {'Data': 'data'}, inplace=True)
```

In [245…

```
milho_df
```

Out[245…

	data	milho_reais	milho_dolares
Data			
2004-08-02	2004-08-02	18.24	5.98
2004-08-03	2004-08-03	18.04	5.91
2004-08-04	2004-08-04	18.02	5.90
2004-08-05	2004-08-05	18.06	5.89
2004-08-06	2004-08-06	18.13	5.98
...	...	...	...
2021-02-25	2021-02-25	85.59	15.55
2021-02-26	2021-02-26	85.41	15.30
2021-03-01	2021-03-01	85.59	15.29
2021-03-02	2021-03-02	86.11	15.20
2021-03-03	2021-03-03	87.06	15.14

4127 rows × 3 columns

In [246…

```
ccmfut_df
```

Out[246…

	data	ccmfut_abertura	ccmfut_máxima	ccmfut_mínima	ccmfut_fechamento	ccmfut_volume_fin
Data						
2008-09-19	2008-09-19	22.64	22.64	22.64	22.64	1129500.0
2008-09-22	2008-09-22	22.64	22.64	22.64	22.64	112950.0
2008-09-26	2008-09-26	22.68	22.68	22.68	22.68	565875.0
2008-09-30	2008-09-30	22.55	22.55	22.55	22.55	112500.0
2008-10-02	2008-10-02	22.64	22.64	22.64	22.64	11295.0
...	...	...	...	...	...	...
2021-02-25	2021-02-25	89.45	89.72	88.81	89.63	106361550.0
2021-02-26	2021-02-26	89.61	89.64	88.86	88.86	69200707.5
2021-03-01	2021-03-01	88.92	89.18	88.00	88.70	133054398.0
2021-03-02	2021-03-02	88.80	89.06	88.54	88.94	95795617.5

	data	ccmfut_abertura	ccmfut_máxima	ccmfut_mínima	ccmfut_fechamento	ccmfut_volume_fin
Data						
2021-03-03	2021-03-03	89.00	91.10	88.86	91.05	211768164.0

2917 rows × 6 columns

In [247…

```
mc_df = ccmfut_df.merge(
    milho_df.set_index('data'), how='left', on='data'
)
```

In [248…

```
mc_df
```

Out[248…

	data	ccmfut_abertura	ccmfut_máxima	ccmfut_mínima	ccmfut_fechamento	ccmfut_volume_fin
0	2008-09-19	22.64	22.64	22.64	22.64	1129500.0
1	2008-09-22	22.64	22.64	22.64	22.64	112950.0
2	2008-09-26	22.68	22.68	22.68	22.68	565875.0
3	2008-09-30	22.55	22.55	22.55	22.55	112500.0
4	2008-10-02	22.64	22.64	22.64	22.64	11295.0
...	...	...	...	...	...	...
2912	2021-02-25	89.45	89.72	88.81	89.63	106361550.0
2913	2021-02-26	89.61	89.64	88.86	88.86	69200707.5
2914	2021-03-01	88.92	89.18	88.00	88.70	133054398.0
2915	2021-03-02	88.80	89.06	88.54	88.94	95795617.5
2916	2021-03-03	89.00	91.10	88.86	91.05	211768164.0

2917 rows × 8 columns

In [249…

```
mc_df.isnull().sum()
```

Out[249…

```
data                 0
ccmfut_abertura      0
ccmfut_máxima        0
ccmfut_mínima        0
ccmfut_fechamento    0
```

```
ccmfut_volume_fin    0
milho_reais          2
milho_dolares        2
dtype: int64
```

In [250...

```
mc_df_mvmilho_reais = mc_df["milho_reais"].rolling(5).mean().shift(-5).round(0)
mc_df_mvmilho_dolares = mc_df["milho_dolares"].rolling(5).mean().shift(-5).round(0)
mc_df["milho_reais"].fillna(mc_df_mvmilho_reais, inplace=True)
mc_df["milho_dolares"].fillna(mc_df_mvmilho_dolares, inplace=True)
```

In [251...

```
mc_df.isnull().sum()
```

Out[251...

```
data                 0
ccmfut_abertura      0
ccmfut_máxima        0
ccmfut_mínima        0
ccmfut_fechamento    0
ccmfut_volume_fin    0
milho_reais          1
milho_dolares        1
dtype: int64
```

In [252...

```
mc_df_mvmilho_reais = mc_df["milho_reais"].rolling(5).mean().shift(-5).round(0)
mc_df_mvmilho_dolares = mc_df["milho_dolares"].rolling(5).mean().shift(-5).round(0)
mc_df["milho_reais"].fillna(mc_df_mvmilho_reais, inplace=True)
mc_df["milho_dolares"].fillna(mc_df_mvmilho_dolares, inplace=True)
```

In [253...

```
mc_df.isnull().sum()
```

Out[253...

```
data                 0
ccmfut_abertura      0
ccmfut_máxima        0
ccmfut_mínima        0
ccmfut_fechamento    0
ccmfut_volume_fin    0
milho_reais          0
milho_dolares        0
dtype: int64
```

In [254...

```
mc_df = mc_df.sort_values(by = ['data'])
mc_df
```

Out[254...

	data	ccmfut_abertura	ccmfut_máxima	ccmfut_mínima	ccmfut_fechamento	ccmfut_volume_fin
0	2008-09-19	22.64	22.64	22.64	22.64	1129500.0
1	2008-09-22	22.64	22.64	22.64	22.64	112950.0
2	2008-09-26	22.68	22.68	22.68	22.68	565875.0
3	2008-09-30	22.55	22.55	22.55	22.55	112500.0
4	2008-10-02	22.64	22.64	22.64	22.64	11295.0
...	...	...	...	...	...	...

	data	ccmfut_abertura	ccmfut_máxima	ccmfut_mínima	ccmfut_fechamento	ccmfut_volume_fin
2912	2021-02-25	89.45	89.72	88.81	89.63	106361550.0
2913	2021-02-26	89.61	89.64	88.86	88.86	69200707.5
2914	2021-03-01	88.92	89.18	88.00	88.70	133054398.0
2915	2021-03-02	88.80	89.06	88.54	88.94	95795617.5
2916	2021-03-03	89.00	91.10	88.86	91.05	211768164.0

2917 rows × 8 columns

In [255…

```
mc_df.index = pd.to_datetime(mc_df.data)
mc_df.index.to_period('D')
mc_df.index
```

Out[255…

```
DatetimeIndex(['2008-09-19', '2008-09-22', '2008-09-26', '2008-09-30',
               '2008-10-02', '2008-10-03', '2008-10-06', '2008-10-07',
               '2008-10-08', '2008-10-09',
               ...
               '2021-02-18', '2021-02-19', '2021-02-22', '2021-02-23',
               '2021-02-24', '2021-02-25', '2021-02-26', '2021-03-01',
               '2021-03-02', '2021-03-03'],
              dtype='datetime64[ns]', name='data', length=2917, freq=None)
```

In [256…

```
mc_df
```

Out[256…

	data	ccmfut_abertura	ccmfut_máxima	ccmfut_mínima	ccmfut_fechamento	ccmfut_volume_fin
data						
2008-09-19	2008-09-19	22.64	22.64	22.64	22.64	1129500.0
2008-09-22	2008-09-22	22.64	22.64	22.64	22.64	112950.0
2008-09-26	2008-09-26	22.68	22.68	22.68	22.68	565875.0
2008-09-30	2008-09-30	22.55	22.55	22.55	22.55	112500.0
2008-10-02	2008-10-02	22.64	22.64	22.64	22.64	11295.0
...	...	...	...	...	...	...
2021-02-25	2021-02-25	89.45	89.72	88.81	89.63	106361550.0
2021-02-26	2021-02-26	89.61	89.64	88.86	88.86	69200707.5

	data	ccmfut_abertura	ccmfut_máxima	ccmfut_mínima	ccmfut_fechamento	ccmfut_volume_fin
data						
2021-03-01	2021-03-01	88.92	89.18	88.00	88.70	133054398.0
2021-03-02	2021-03-02	88.80	89.06	88.54	88.94	95795617.5
2021-03-03	2021-03-03	89.00	91.10	88.86	91.05	211768164.0

2917 rows × 8 columns

In [257…

```
# 3 - Analise e Exploracao dos Dados
```

In [258…

```
milho_df.describe()
```

Out[258…

	milho_reais	milho_dolares
count	4127.000000	4127.000000
mean	30.409537	11.535544
std	12.350048	3.276651
min	13.320000	5.890000
25%	21.325000	9.250000
50%	27.770000	10.720000
75%	35.375000	13.750000
max	87.060000	19.960000

In [259…

```
milho_df[milho_df['milho_reais']==milho_df['milho_reais'].max()]
```

Out[259…

	data	milho_reais	milho_dolares
Data			
2021-03-03	2021-03-03	87.06	15.14

In [260…

```
milho_df[milho_df['milho_dolares']==milho_df['milho_dolares'].max()]
```

Out[260…

	data	milho_reais	milho_dolares
Data			
2011-07-01	2011-07-01	31.08	19.96

In [261…

```
milho_df[milho_df['milho_reais']==milho_df['milho_reais'].min()]
```

Out[261…

	data	milho_reais	milho_dolares

Data	data	milho_reais	milho_dolares
Data			
2006-03-30	2006-03-30	13.32	6.08

In [262…

```
milho_df[milho_df['milho_dolares']==milho_df['milho_dolares'].min()]
```

Out[262…

	data	milho_reais	milho_dolares
Data			
2004-08-05	2004-08-05	18.06	5.89

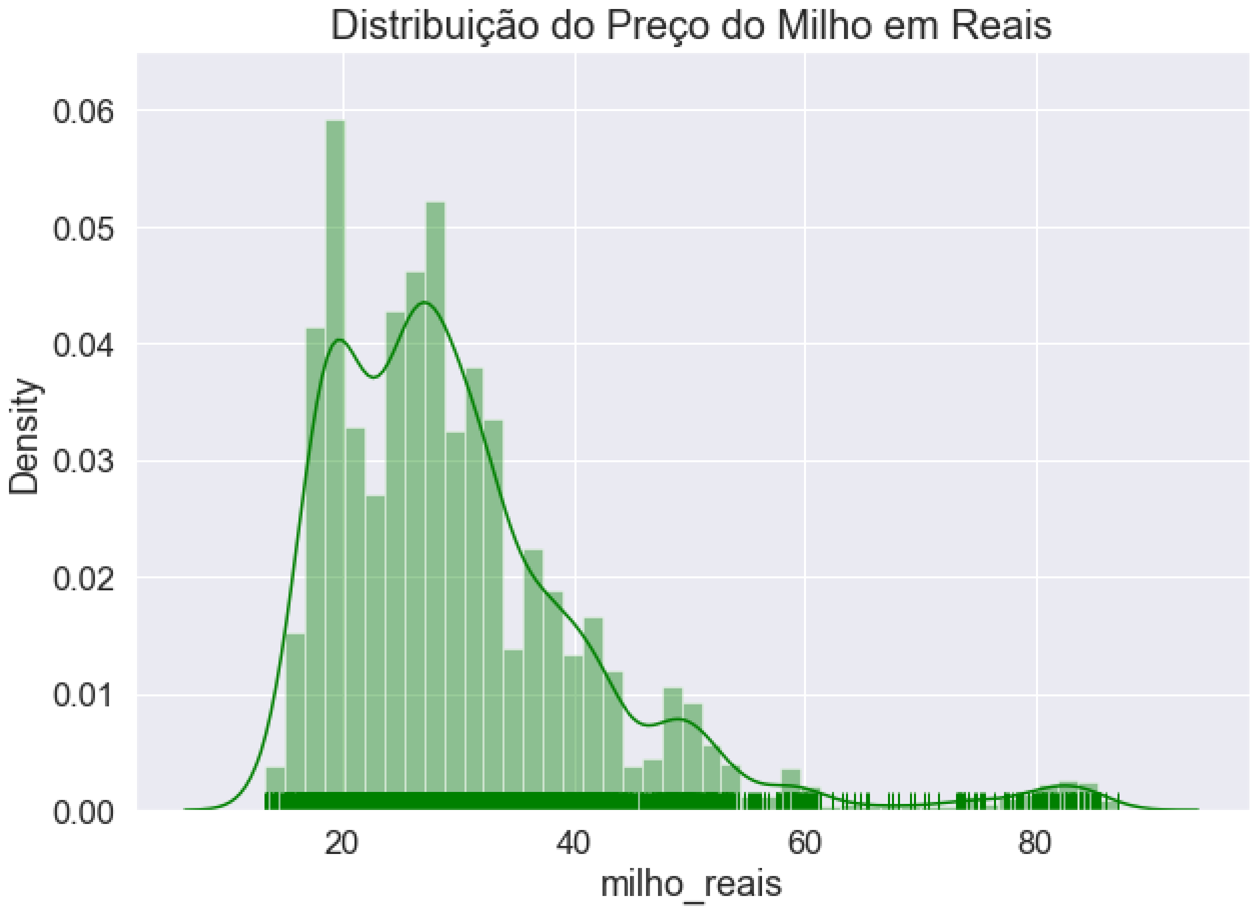
In [263…

```
plt.figure(figsize=(10,7))
sns.set_context('notebook', font_scale=1.5, rc={'font.size':20, 'axes.titlesize':20, 'a
sns.distplot(milho_df['milho_reais'], rug=True, color= 'green')
sns.set_style('darkgrid')
plt.title('Distribuição do Preço do Milho em Reais')
```

```
C:\Users\Jonathan Lincher\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fut
ureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).

C:\Users\Jonathan Lincher\anaconda3\lib\site-packages\seaborn\distributions.py:2055: Fut
ureWarning:

The `axis` variable is no longer used and will be removed. Instead, assign variables dir
ectly to `x` or `y`.
```

Out[263…

```
Text(0.5, 1.0, 'Distribuição do Preço do Milho em Reais')
```



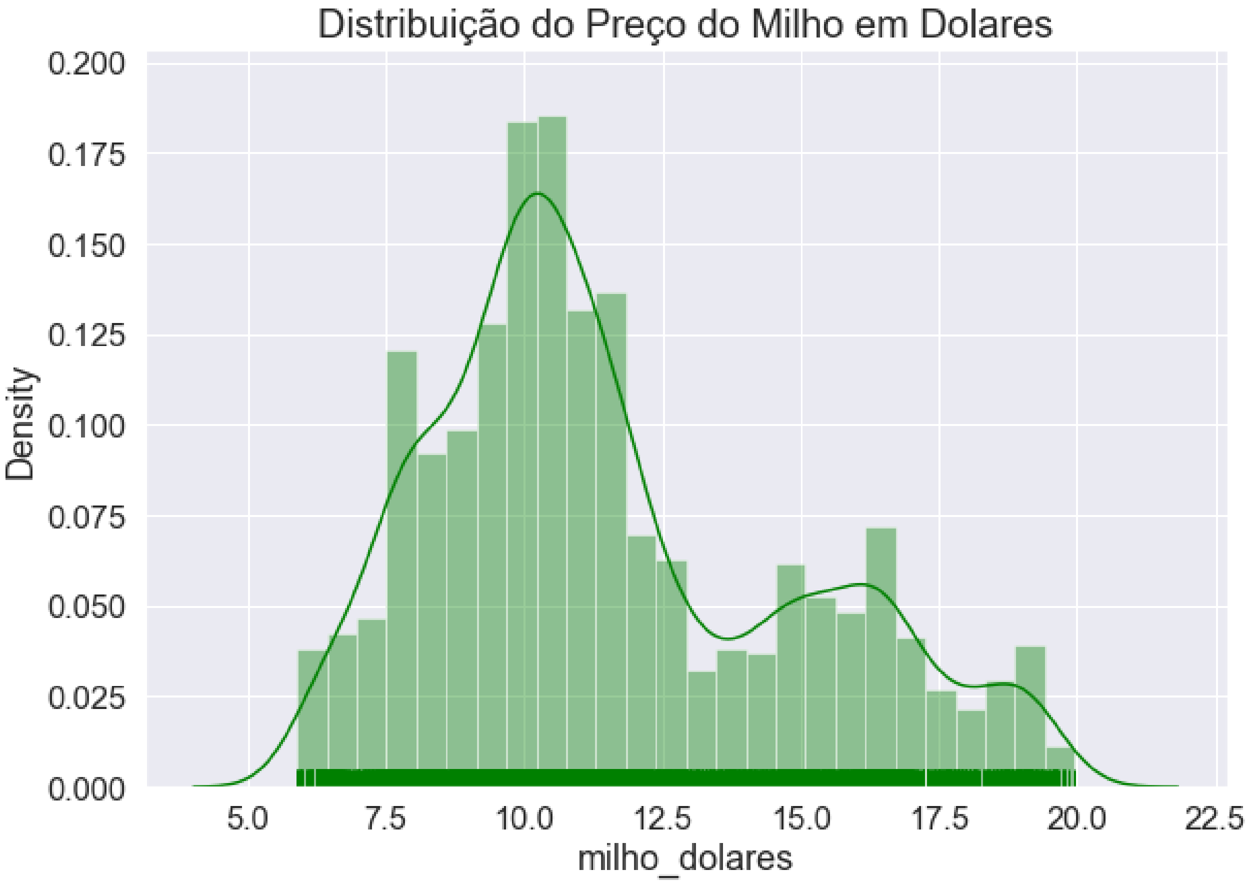
In [264…

```
plt.figure(figsize=(10,7))
sns.set_context('notebook', font_scale=1.5, rc={'font.size':20, 'axes.titlesize':20, 'a
sns.distplot(milho_df['milho_dolares'], rug=True, color= 'green')
sns.set_style('darkgrid')
plt.title('Distribuição do Preço do Milho em Dolares')
```

```
C:\Users\Jonathan Lincher\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fut
ureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).

C:\Users\Jonathan Lincher\anaconda3\lib\site-packages\seaborn\distributions.py:2055: Fut
ureWarning:

The `axis` variable is no longer used and will be removed. Instead, assign variables dir
ectly to `x` or `y`.
```

Out[264…

```
Text(0.5, 1.0, 'Distribuição do Preço do Milho em Dolares')
```



In [265…

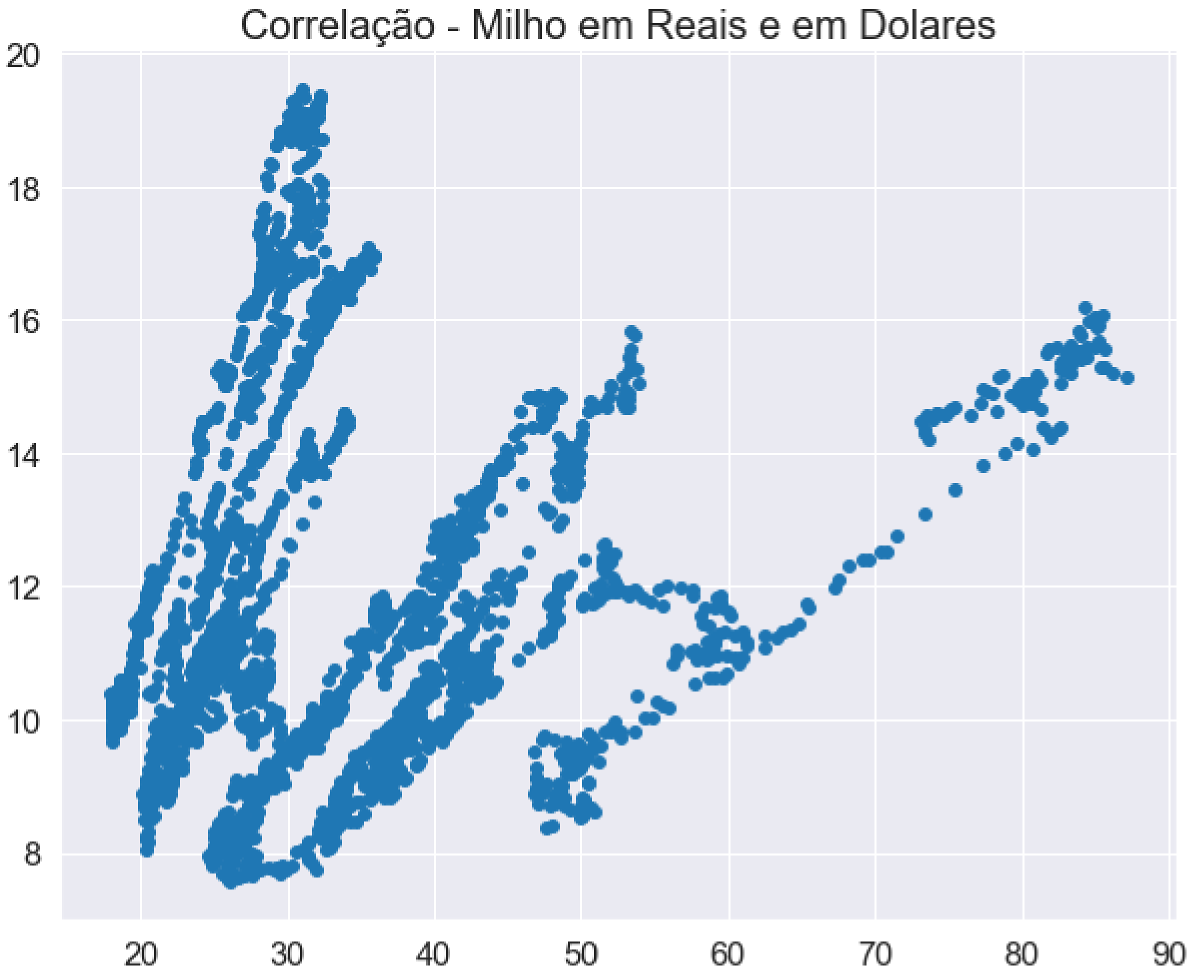
```
mc_df['milho_reais'].corr(mc_df['milho_dolares'])
```

Out[265…

```
0.1816166279649749
```

In [266…

```
data1=mc_df['milho_reais']
data2=mc_df['milho_dolares']
plt.scatter(data1, data2)
plt.title('Correlação - Milho em Reais e em Dolares')
plt.gcf().set_size_inches(10, 8)
plt.show()
```



In [267…

```
def interactive_plot(df, title):
  fig = px.line(title = title)
  for i in df.columns[1:]:
    fig.add_scatter(x = df['data'], y = df[i], name = i)
  fig.show()
```

In [268…

```
interactive_plot(milho_df, "Milho em Reais e Milho em Dolares")
```

In [269…

```python
dolar_df = pd.read_csv('USD_BRL Dados Históricos.csv', decimal=",")
dolar_df
```

Out[269…

	Data	Último	Abertura	Máxima	Mínima	Var%
0	03.03.2021	5.6193	5.6872	5.7729	5.5806	-1,01%
1	02.03.2021	5.6764	5.6386	5.7327	5.6386	0,61%
2	01.03.2021	5.6418	5.5870	5.6427	5.5553	0,77%
3	26.02.2021	5.5986	5.5340	5.6093	5.4905	1,23%
4	25.02.2021	5.5308	5.4450	5.5390	5.4173	2,30%
...	...	...	...	...	...	...
4319	06.08.2004	3.0330	3.0722	3.0780	3.0300	-1,25%
4320	05.08.2004	3.0713	3.0540	3.0713	3.0500	0,58%
4321	04.08.2004	3.0537	3.0500	3.0660	3.0460	0,12%
4322	03.08.2004	3.0500	3.0450	3.0620	3.0440	0,11%
4323	02.08.2004	3.0465	3.0365	3.0585	3.0365	0,30%

4324 rows × 6 columns

In [270…

```python
dolar_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4324 entries, 0 to 4323
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Data      4324 non-null   object 
 1   Último    4324 non-null   float64
 2   Abertura  4324 non-null   float64
 3   Máxima    4324 non-null   float64
 4   Mínima    4324 non-null   float64
 5   Var%      4324 non-null   object 
dtypes: float64(4), object(2)
memory usage: 202.8+ KB
```

In [271…

```python
dolar_df = dolar_df.drop(columns=dolar_df.columns[2:])
dolar_df
```

Out[271…

	Data	Último

	Data	Último
0	03.03.2021	5.6193
1	02.03.2021	5.6764
2	01.03.2021	5.6418
3	26.02.2021	5.5986
4	25.02.2021	5.5308
...	...	...
4319	06.08.2004	3.0330
4320	05.08.2004	3.0713
4321	04.08.2004	3.0537
4322	03.08.2004	3.0500
4323	02.08.2004	3.0465

4324 rows × 2 columns

In [272...

```
dolar_df ['Data'] = pd.to_datetime(dolar_df ['Data'],dayfirst=True)
dolar_df = dolar_df.sort_values(by = ['Data'])
dolar_df['Data']
```

Out[272...

```
4323   2004-08-02
4322   2004-08-03
4321   2004-08-04
4320   2004-08-05
4319   2004-08-06
          ...
4      2021-02-25
3      2021-02-26
2      2021-03-01
1      2021-03-02
0      2021-03-03
Name: Data, Length: 4324, dtype: datetime64[ns]
```

In [273...

```
dolar_df.index = pd.to_datetime(dolar_df.Data)
dolar_df.index.to_period('D')
dolar_df.index
```

Out[273...

```
DatetimeIndex(['2004-08-02', '2004-08-03', '2004-08-04', '2004-08-05',
               '2004-08-06', '2004-08-09', '2004-08-10', '2004-08-11',
               '2004-08-12', '2004-08-13',
               ...
               '2021-02-18', '2021-02-19', '2021-02-22', '2021-02-23',
               '2021-02-24', '2021-02-25', '2021-02-26', '2021-03-01',
               '2021-03-02', '2021-03-03'],
              dtype='datetime64[ns]', name='Data', length=4324, freq=None)
```

In [274...

```
dolar_df.rename(columns= {'Data': 'data'}, inplace=True)
dolar_df.rename(columns= {'Último': 'Dolar_Último'}, inplace=True)
```

In [275...

```
dolar_df
```

Out[275...

	data	Dolar_Último

Data	data	Dolar_Último
Data		
2004-08-02	2004-08-02	3.0465
2004-08-03	2004-08-03	3.0500
2004-08-04	2004-08-04	3.0537
2004-08-05	2004-08-05	3.0713
2004-08-06	2004-08-06	3.0330
...	...	...
2021-02-25	2021-02-25	5.5308
2021-02-26	2021-02-26	5.5986
2021-03-01	2021-03-01	5.6418
2021-03-02	2021-03-02	5.6764
2021-03-03	2021-03-03	5.6193

4324 rows × 2 columns

In [276…

```
milho_df
```

Out[276…

	data	milho_reais	milho_dolares
Data			
2004-08-02	2004-08-02	18.24	5.98
2004-08-03	2004-08-03	18.04	5.91
2004-08-04	2004-08-04	18.02	5.90
2004-08-05	2004-08-05	18.06	5.89
2004-08-06	2004-08-06	18.13	5.98
...	...	...	...
2021-02-25	2021-02-25	85.59	15.55
2021-02-26	2021-02-26	85.41	15.30
2021-03-01	2021-03-01	85.59	15.29
2021-03-02	2021-03-02	86.11	15.20
2021-03-03	2021-03-03	87.06	15.14

4127 rows × 3 columns

In [277…

```
dolar_milho_df = pd.merge(milho_df,dolar_df, how='inner', on=['Data'],suffixes=('_M', '
dolar_milho_df
```

Out[277…

	data_M	milho_reais	milho_dolares	data_D	Dolar_Último

	data_M	milho_reais	milho_dolares	data_D	Dolar_Último
Data					
2004-08-02	2004-08-02	18.24	5.98	2004-08-02	3.0465
2004-08-03	2004-08-03	18.04	5.91	2004-08-03	3.0500
2004-08-04	2004-08-04	18.02	5.90	2004-08-04	3.0537
2004-08-05	2004-08-05	18.06	5.89	2004-08-05	3.0713
2004-08-06	2004-08-06	18.13	5.98	2004-08-06	3.0330
...	...	...	...	...	...
2021-02-25	2021-02-25	85.59	15.55	2021-02-25	5.5308
2021-02-26	2021-02-26	85.41	15.30	2021-02-26	5.5986
2021-03-01	2021-03-01	85.59	15.29	2021-03-01	5.6418
2021-03-02	2021-03-02	86.11	15.20	2021-03-02	5.6764
2021-03-03	2021-03-03	87.06	15.14	2021-03-03	5.6193

4127 rows × 5 columns

In [278…

```
dolar_milho_df = dolar_milho_df.drop(columns='data_D')
dolar_milho_df
```

Out[278…

	data_M	milho_reais	milho_dolares	Dolar_Último
Data				
2004-08-02	2004-08-02	18.24	5.98	3.0465
2004-08-03	2004-08-03	18.04	5.91	3.0500
2004-08-04	2004-08-04	18.02	5.90	3.0537
2004-08-05	2004-08-05	18.06	5.89	3.0713
2004-08-06	2004-08-06	18.13	5.98	3.0330
...	...	...	...	...
2021-02-25	2021-02-25	85.59	15.55	5.5308
2021-02-26	2021-02-26	85.41	15.30	5.5986
2021-03-01	2021-03-01	85.59	15.29	5.6418
2021-03-02	2021-03-02	86.11	15.20	5.6764
2021-03-03	2021-03-03	87.06	15.14	5.6193

4127 rows × 4 columns

In [279…

```
dolar_milho_df['milho_reais'].corr(dolar_milho_df['Dolar_Último'])
```

Out[279…

```
0.7824021362013128
```

In [280…

```
ccmfut_df.describe()
```

Out[280…

	ccmfut_abertura	ccmfut_máxima	ccmfut_mínima	ccmfut_fechamento	ccmfut_volume_fin
count	2917.000000	2917.000000	2917.000000	2917.000000	2.917000e+03
mean	26.723809	26.988920	26.474443	26.742749	3.300606e+07
std	13.506372	13.711821	13.312827	13.540517	4.274579e+07
min	12.680000	12.800000	12.310000	12.710000	0.000000e+00
25%	19.620000	19.780000	19.500000	19.640000	1.002070e+07
50%	22.560000	22.760000	22.350000	22.560000	2.244742e+07
75%	30.140000	30.510000	29.820000	30.140000	3.881796e+07
max	89.730000	91.120000	89.280000	91.050000	6.528013e+08

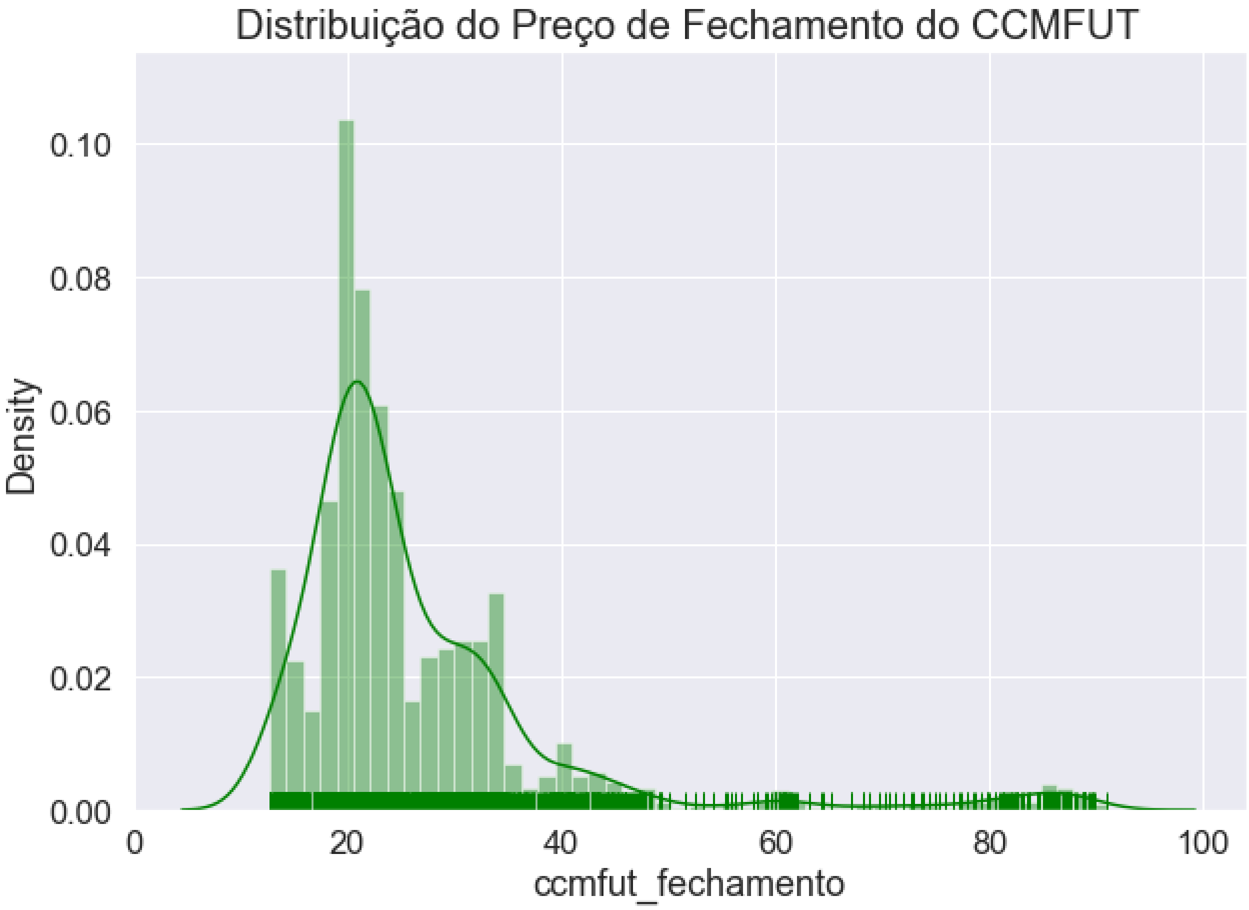
In [281…

```
plt.figure(figsize=(10,7))
sns.set_context('notebook', font_scale=1.5, rc={'font.size':20, 'axes.titlesize':20, 'a
sns.distplot(ccmfut_df['ccmfut_fechamento'], rug=True, color= 'green')
sns.set_style('darkgrid')
plt.title('Distribuição do Preço de Fechamento do CCMFUT')
```

```
C:\Users\Jonathan Lincher\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fut
ureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).

C:\Users\Jonathan Lincher\anaconda3\lib\site-packages\seaborn\distributions.py:2055: Fut
ureWarning:

The `axis` variable is no longer used and will be removed. Instead, assign variables dir
ectly to `x` or `y`.
```

Out[281…

```
Text(0.5, 1.0, 'Distribuição do Preço de Fechamento do CCMFUT')
```



In [282…

```
def interactive_plot(df, title):
  fig = px.line(title = title)
  for i in df.columns[1:]:
    fig.add_scatter(x = df['data'], y = df[i], name = i)
  fig.show()
```

In [283…

```
interactive_plot(ccmfut_df, "Historico de Precos - CCMFUT")
```

In [284…

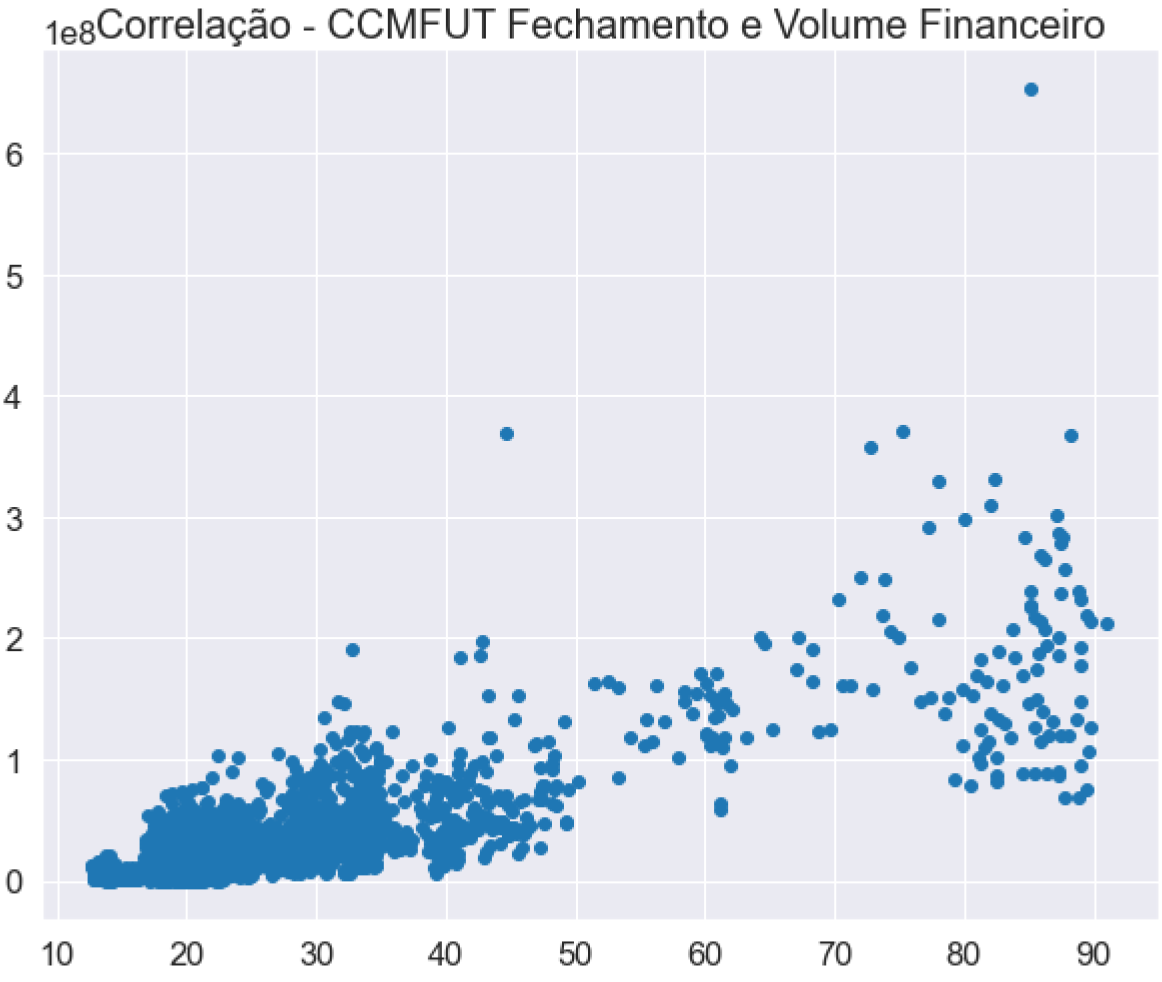
```
ccmfut_df['ccmfut_fechamento'].corr(ccmfut_df['ccmfut_volume_fin'])
```

Out[284…

```
0.7999203525936038
```

In [285…

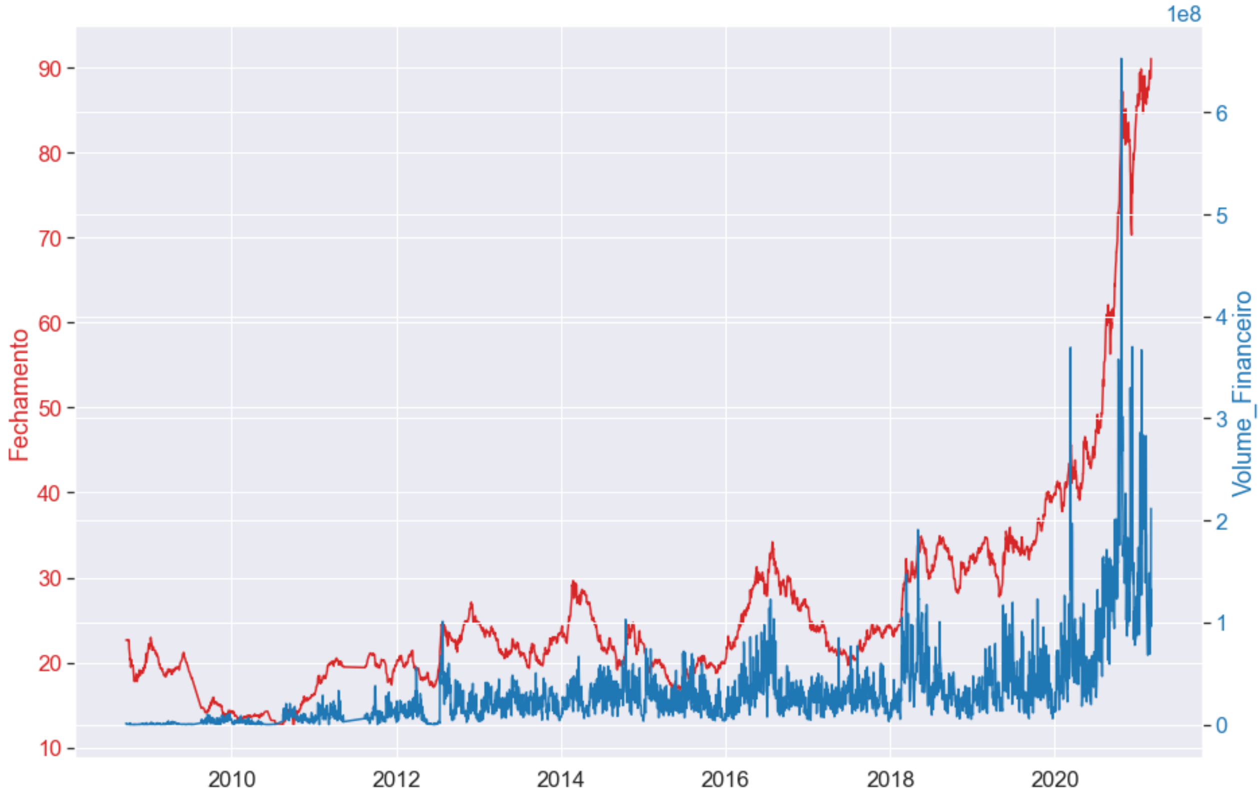
```
data1=ccmfut_df['ccmfut_fechamento']
data2=ccmfut_df['ccmfut_volume_fin']
plt.scatter(data1, data2)
plt.title('Correlação - CCMFUT Fechamento e Volume Financeiro')
plt.gcf().set_size_inches(10, 8)
plt.show()
```



In [286…

```
b = ccmfut_df["data"]
data1 = ccmfut_df["ccmfut_fechamento"]
data2 = ccmfut_df["ccmfut_volume_fin"]

fig, ax1 = plt.subplots()
```

```
color = 'tab:red'
ax1.set_xlabel('')
ax1.set_ylabel('Fechamento', color=color)
ax1.plot(b, data1, color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()

color = 'tab:blue'
ax2.set_ylabel('Volume_Financeiro', color=color)
ax2.plot(b, data2, color=color)
ax2.tick_params(axis='y', labelcolor=color)
plt.gcf().set_size_inches(15, 10)
plt.show()
```



In [287…

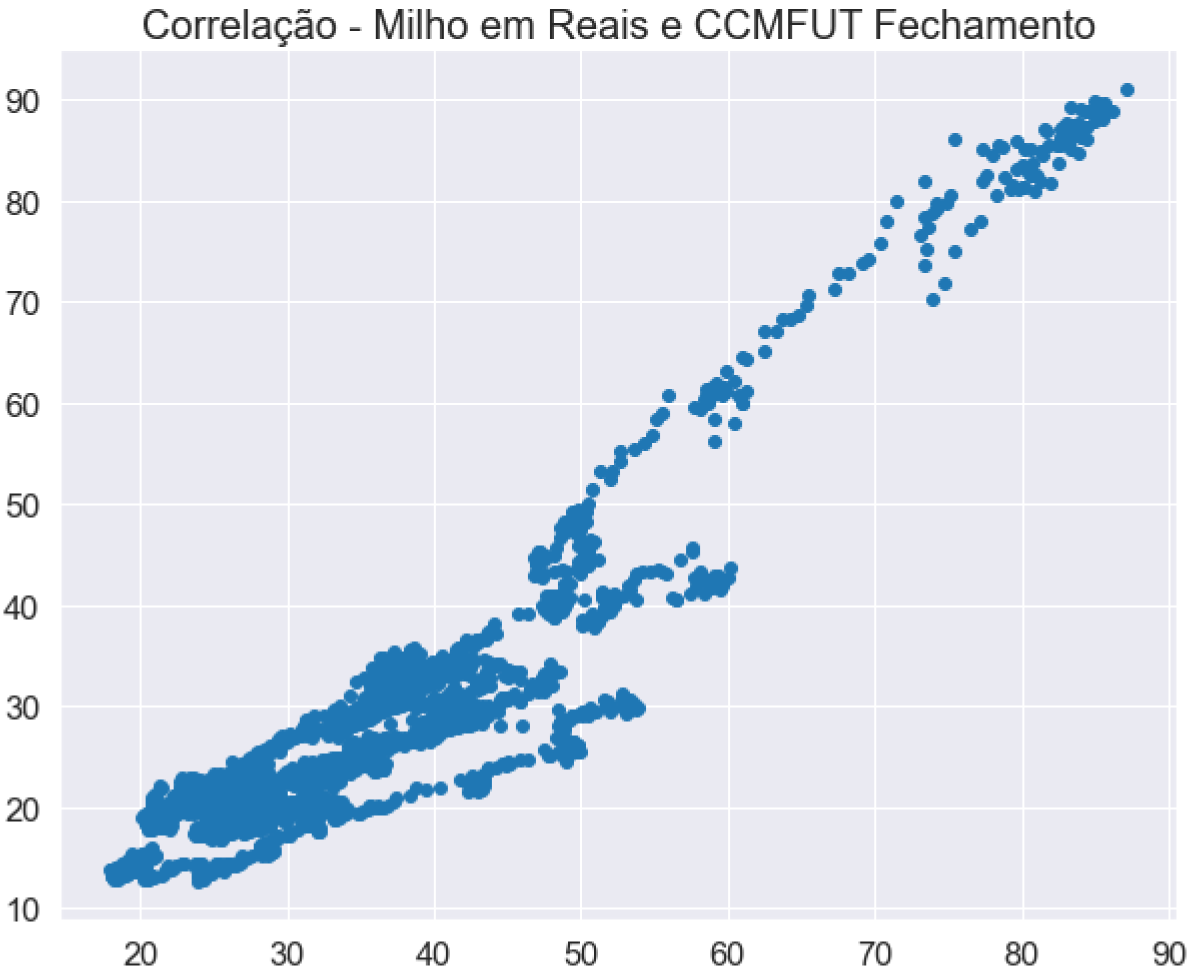
```
mc_df['milho_reais'].corr(mc_df['ccmfut_fechamento'])
```

Out[287…

```
0.928074900682255
```

In [288…

```
data1=mc_df['milho_reais']
data2=mc_df['ccmfut_fechamento']
plt.scatter(data1, data2)
plt.title('Correlação - Milho em Reais e CCMFUT Fechamento')
plt.gcf().set_size_inches(10, 8)
plt.show()
```



In [289…

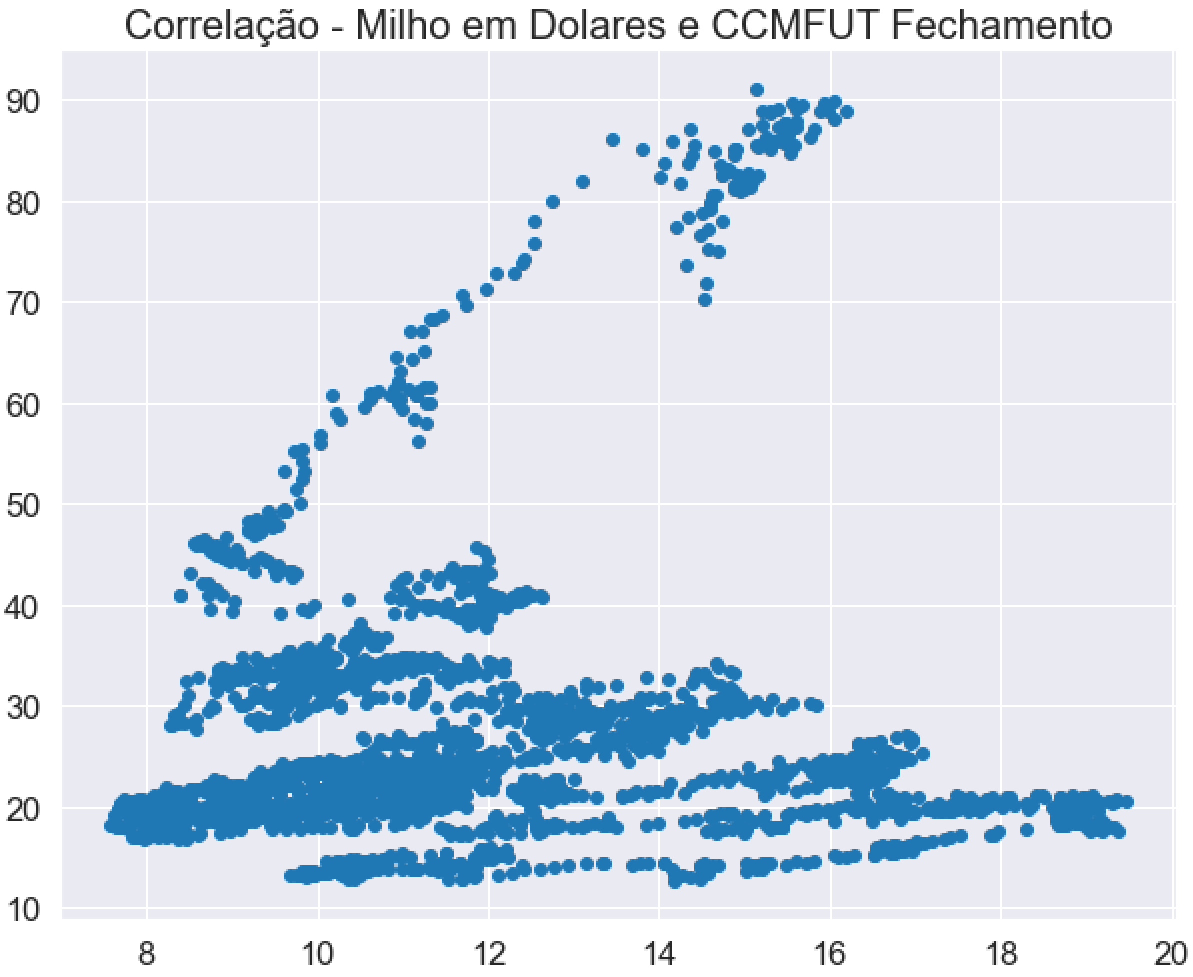
```
mc_df['milho_dolares'].corr(mc_df['ccmfut_fechamento'])
```

Out[289…

```
0.065512943089332
```

In [290…

```
data1=mc_df['milho_dolares']
data2=mc_df['ccmfut_fechamento']
plt.scatter(data1, data2)
plt.title('Correlação - Milho em Dolares e CCMFUT Fechamento')
plt.gcf().set_size_inches(10, 8)
plt.show()
```



Correlação - Milho em Dolares e CCMFUT Fechamento

In [291…

```
mc_df = mc_df[['data', 'ccmfut_abertura', 'ccmfut_máxima', 'ccmfut_mínima' , 'ccmfut_vo
```

In [292…

```
def interactive_plot(df, title):
  fig = px.line(title = title)
  for i in df.columns[5:]:
    fig.add_scatter(x = df['data'], y = df[i], name = i)
  fig.show()
```

In [293…

```
interactive_plot(mc_df, "CCMFUT Fechamento, Milho em Reais e Milho em Dolares")
```

In [294...

```
# 4 - Ridge Regression
```

In [295...

```
mc_df['ccmfut_fechamento_alvo'] = mc_df[['ccmfut_fechamento']].shift(-1)
mc_df = mc_df[:-1]
mc_df
```

Out[295...

	data	ccmfut_abertura	ccmfut_máxima	ccmfut_mínima	ccmfut_volume_fin	ccmfut_fechamento
data						
2008-09-19	2008-09-19	22.64	22.64	22.64	1129500.0	22.64
2008-09-22	2008-09-22	22.64	22.64	22.64	112950.0	22.64
2008-09-26	2008-09-26	22.68	22.68	22.68	565875.0	22.68
2008-09-30	2008-09-30	22.55	22.55	22.55	112500.0	22.55
2008-10-02	2008-10-02	22.64	22.64	22.64	11295.0	22.64
...	...	...	...	...	...	...
2021-02-24	2021-02-24	89.08	89.47	88.42	75830647.5	89.47
2021-02-25	2021-02-25	89.45	89.72	88.81	106361550.0	89.63
2021-02-26	2021-02-26	89.61	89.64	88.86	69200707.5	88.86
2021-03-01	2021-03-01	88.92	89.18	88.00	133054398.0	88.70
2021-03-02	2021-03-02	88.80	89.06	88.54	95795617.5	88.94

2916 rows × 9 columns

In [296...

```
from sklearn.preprocessing import MinMaxScaler
```

```
sc = MinMaxScaler(feature_range = (0, 1))
mc_df_scaled = sc.fit_transform(mc_df.drop(columns = ['data']))
```

In [297…

```
mc_df_scaled
```

Out[297…

```
array([[0.12926671, 0.12563841, 0.13420813, ..., 0.08071617, 0.44107744,
        0.12675517],
       [0.12926671, 0.12563841, 0.13420813, ..., 0.07836807, 0.45622896,
        0.12726576],
       [0.12978585, 0.12614913, 0.13472782, ..., 0.07734077, 0.41750842,
        0.12560633],
       ...,
       [0.99844257, 0.98110317, 0.99454333, ..., 0.98972703, 0.64983165,
        0.97000255],
       [0.98948735, 0.97522983, 0.98337014, ..., 0.99236865, 0.6489899 ,
        0.97306612],
       [0.98792992, 0.97369765, 0.99038586, ..., 1.        , 0.64141414,
        1.        ]])
```

In [298…

```
mc_df_scaled.shape
```

Out[298…

```
(2916, 8)
```

In [299…

```
X = mc_df_scaled[:,:7]
y = mc_df_scaled[:,7:]
```

In [300…

```
X
```

Out[300…

```
array([[0.12926671, 0.12563841, 0.13420813, ..., 0.12877707, 0.08071617,
        0.44107744],
       [0.12926671, 0.12563841, 0.13420813, ..., 0.12877707, 0.07836807,
        0.45622896],
       [0.12978585, 0.12614913, 0.13472782, ..., 0.12929581, 0.07734077,
        0.41750842],
       ...,
       [0.99844257, 0.98110317, 0.99454333, ..., 0.98755025, 0.98972703,
        0.64983165],
       [0.98948735, 0.97522983, 0.98337014, ..., 0.9854753 , 0.99236865,
        0.6489899 ],
       [0.98792992, 0.97369765, 0.99038586, ..., 0.98858773, 1.        ,
        0.64141414]])
```

In [301…

```
X.shape
```

Out[301…

```
(2916, 7)
```

In [302…

```
X[0,6]
```

Out[302…

```
0.44107744107744096
```

In [303…

```
y
```

Out[303…

```
array([[0.12675517],
       [0.12726576],
       [0.12560633],
       ...,
       [0.97000255],
       [0.97306612],
       [1.        ]])
```

In [304…

```
X = np.asarray(X)
y = np.asarray(y)
X.shape, y.shape
```

Out[304…

```
((2916, 7), (2916, 1))
```

In [305…

```
X
```

Out[305…

```
array([[0.12926671, 0.12563841, 0.13420813, ..., 0.12877707, 0.08071617,
        0.44107744],
       [0.12926671, 0.12563841, 0.13420813, ..., 0.12877707, 0.07836807,
        0.45622896],
       [0.12978585, 0.12614913, 0.13472782, ..., 0.12929581, 0.07734077,
        0.41750842],
       ...,
       [0.99844257, 0.98110317, 0.99454333, ..., 0.98755025, 0.98972703,
        0.64983165],
       [0.98948735, 0.97522983, 0.98337014, ..., 0.9854753 , 0.99236865,
        0.6489899 ],
       [0.98792992, 0.97369765, 0.99038586, ..., 0.98858773, 1.        ,
        0.64141414]])
```

In [306…

```
y
```

Out[306…

```
array([[0.12675517],
       [0.12726576],
       [0.12560633],
       ...,
       [0.97000255],
       [0.97306612],
       [1.        ]])
```

In [307…

```
split = int(0.70 * len(X))
X_treino = X[:split]
y_treino = y[:split]
X_teste = X[split:]
y_teste = y[split:]
```

In [308…

```
X_treino.shape, y_treino.shape
```

Out[308…

```
((2041, 7), (2041, 1))
```

In [309…

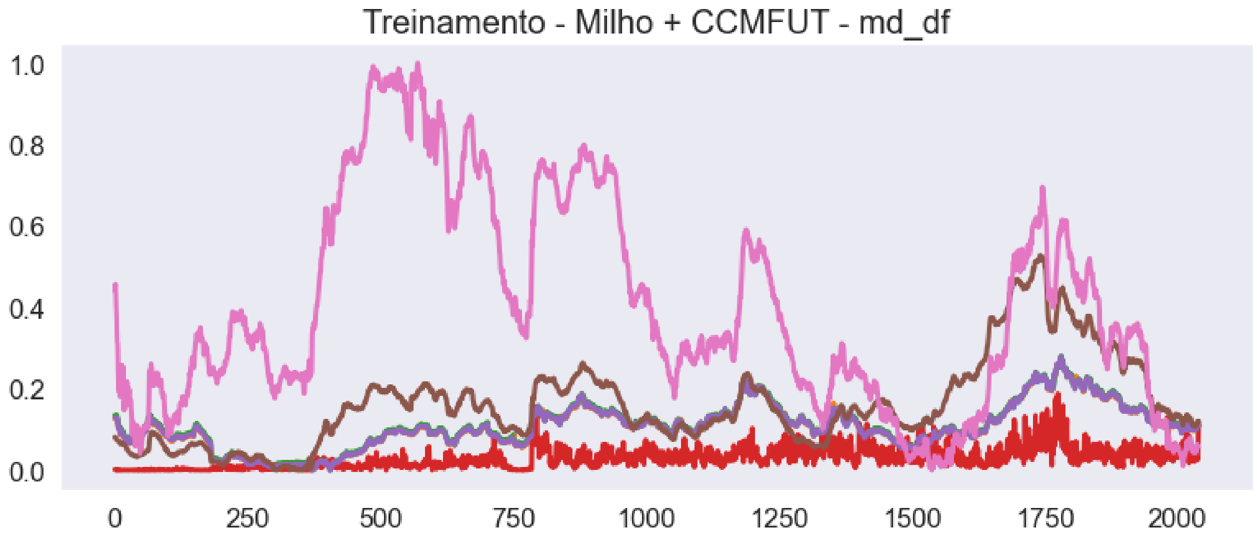
```
X_teste.shape, y_teste.shape
```

Out[309…

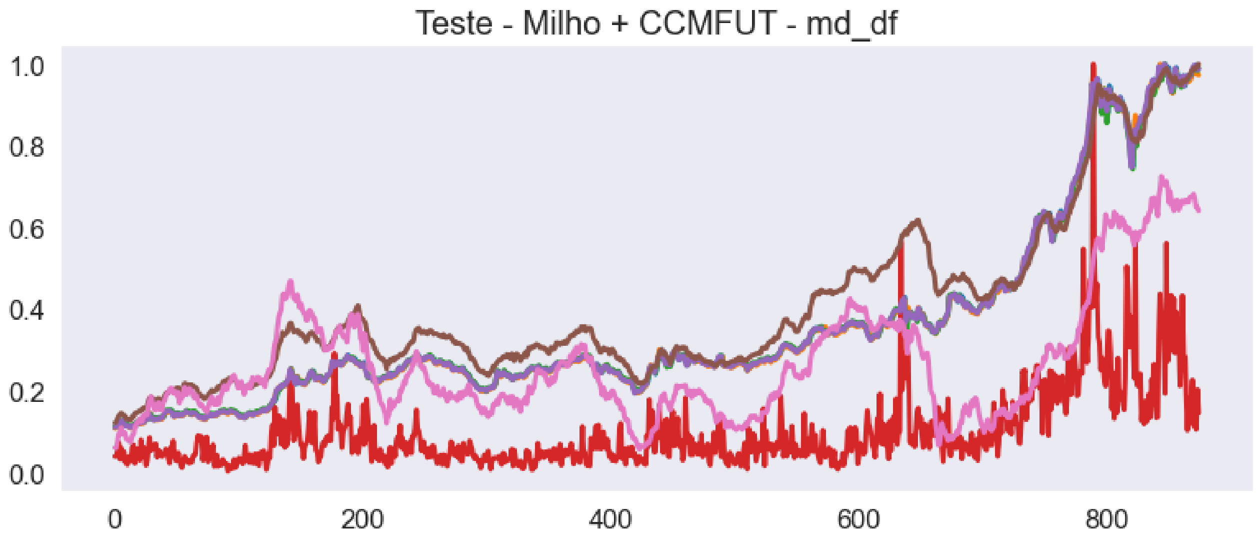
```
((875, 7), (875, 1))
```

In [310…

```
def show_plot_mc(data, title):
  plt.figure(figsize = (13, 5))
  plt.plot(data, linewidth = 3)
  plt.title(title)
  plt.grid()

show_plot_mc(X_treino, 'Treinamento - Milho + CCMFUT - md_df')
show_plot_mc(X_teste, 'Teste - Milho + CCMFUT - md_df')
```



Treinamento - Milho + CCMFUT - md_df



Teste - Milho + CCMFUT - md_df

```
In [311…  from sklearn.linear_model import Ridge
          regression_model = Ridge (alpha=1)
          regression_model.fit(X_treino, y_treino)

Out[311… Ridge(alpha=1)
```

```
In [312…  lr_accuracy = regression_model.score(X_teste, y_teste)
          print("Linear Regression Score: ", lr_accuracy)

          Linear Regression Score:  0.9874692042284041
```

```
In [313…  predicted_prices_mc = regression_model.predict(X)
          predicted_prices_mc

Out[313… array([[0.12060122],
                [0.12045201],
                [0.12093868],
                ...,
                [0.91668274],
                [0.91338118],
                [0.91366199]])
```

```
In [314…  predicted_mc = []
          for i in predicted_prices_mc:
            predicted_mc.append(i[0])
```

In [315…

```
mc_fechamento = []
for i in mc_df_scaled:
  mc_fechamento.append(i[4])
```

In [316…

```
mc_predicao = pd.DataFrame(columns = ['data' , 'mc_fechamento', 'mc_fechamento_predito'
mc_predicao['data'] = mc_df['data']
mc_predicao['mc_fechamento'] = mc_fechamento
mc_predicao['mc_fechamento_predito'] = predicted_mc
mc_predicao
```

Out[316…

	data	mc_fechamento	mc_fechamento_predito
data			
2008-09-19	2008-09-19	0.128777	0.120601
2008-09-22	2008-09-22	0.128777	0.120452
2008-09-26	2008-09-26	0.129296	0.120939
2008-09-30	2008-09-30	0.127610	0.119349
2008-10-02	2008-10-02	0.128777	0.120427
...	...	...	...
2021-02-24	2021-02-24	0.995461	0.915434
2021-02-25	2021-02-25	0.997536	0.920280
2021-02-26	2021-02-26	0.987550	0.916683
2021-03-01	2021-03-01	0.985475	0.913381
2021-03-02	2021-03-02	0.988588	0.913662

2916 rows × 3 columns

In [317…

```
def interactive_plot(data, title):
  fig = px.line(title = title)
  for i in data.columns[1:]:
    fig.add_scatter(x = data['data'], y = data[i], name = i)
  fig.show()
```

In [318…

```
interactive_plot(mc_predicao, "CCMFUT Fechamento e CCMFUT Fechamento Predito")
```

In [319…

```
#Cálculo do erro
mse = mean_squared_error(mc_predicao['mc_fechamento'], mc_predicao['mc_fechamento_predi
print('MSE: '+str(mse))
mae = mean_absolute_error(mc_predicao['mc_fechamento'], mc_predicao['mc_fechamento_pred
print('MAE: '+str(mae))
rmse = math.sqrt(mean_squared_error(mc_predicao['mc_fechamento'], mc_predicao['mc_fecha
print('RMSE: '+str(rmse))
```

```
MSE: 0.0002551720660963173
MAE: 0.008955258899856846
RMSE: 0.015974106112591004
```

In [320…

```
mc_df
```

Out[320…

	data	ccmfut_abertura	ccmfut_máxima	ccmfut_mínima	ccmfut_volume_fin	ccmfut_fechamento
data						
2008-09-19	2008-09-19	22.64	22.64	22.64	1129500.0	22.64
2008-09-22	2008-09-22	22.64	22.64	22.64	112950.0	22.64
2008-09-26	2008-09-26	22.68	22.68	22.68	565875.0	22.68
2008-09-30	2008-09-30	22.55	22.55	22.55	112500.0	22.55
2008-10-02	2008-10-02	22.64	22.64	22.64	11295.0	22.64
...	...	...	...	...	...	...
2021-02-24	2021-02-24	89.08	89.47	88.42	75830647.5	89.47
2021-02-25	2021-02-25	89.45	89.72	88.81	106361550.0	89.63
2021-02-26	2021-02-26	89.61	89.64	88.86	69200707.5	88.86

	data	ccmfut_abertura	ccmfut_máxima	ccmfut_mínima	ccmfut_volume_fin	ccmfut_fechamento
data						
2021-03-01	2021-03-01	88.92	89.18	88.00	133054398.0	88.70
2021-03-02	2021-03-02	88.80	89.06	88.54	95795617.5	88.94

2916 rows × 9 columns

In [321…

```
mc_df = mc_df.drop(columns='ccmfut_abertura')
mc_df = mc_df.drop(columns='ccmfut_máxima')
mc_df = mc_df.drop(columns='ccmfut_mínima')
mc_df = mc_df.drop(columns='ccmfut_volume_fin')
mc_df = mc_df.drop(columns='milho_reais')
mc_df = mc_df.drop(columns='milho_dolares')
mc_df
```

Out[321…

	data	ccmfut_fechamento	ccmfut_fechamento_alvo
data			
2008-09-19	2008-09-19	22.64	22.64
2008-09-22	2008-09-22	22.64	22.68
2008-09-26	2008-09-26	22.68	22.55
2008-09-30	2008-09-30	22.55	22.64
2008-10-02	2008-10-02	22.64	21.65
...	...	...	...
2021-02-24	2021-02-24	89.47	89.63
2021-02-25	2021-02-25	89.63	88.86
2021-02-26	2021-02-26	88.86	88.70
2021-03-01	2021-03-01	88.70	88.94
2021-03-02	2021-03-02	88.94	91.05

2916 rows × 3 columns

In [322…

```
mc_df_scaled_fech = sc.fit_transform(mc_df.drop(columns = ['data']))
```

In [323…

```
X = mc_df_scaled_fech[:,:1]
y = mc_df_scaled_fech[:,1:]
```

In [324…

```
X = np.asarray(X)
y = np.asarray(y)
```

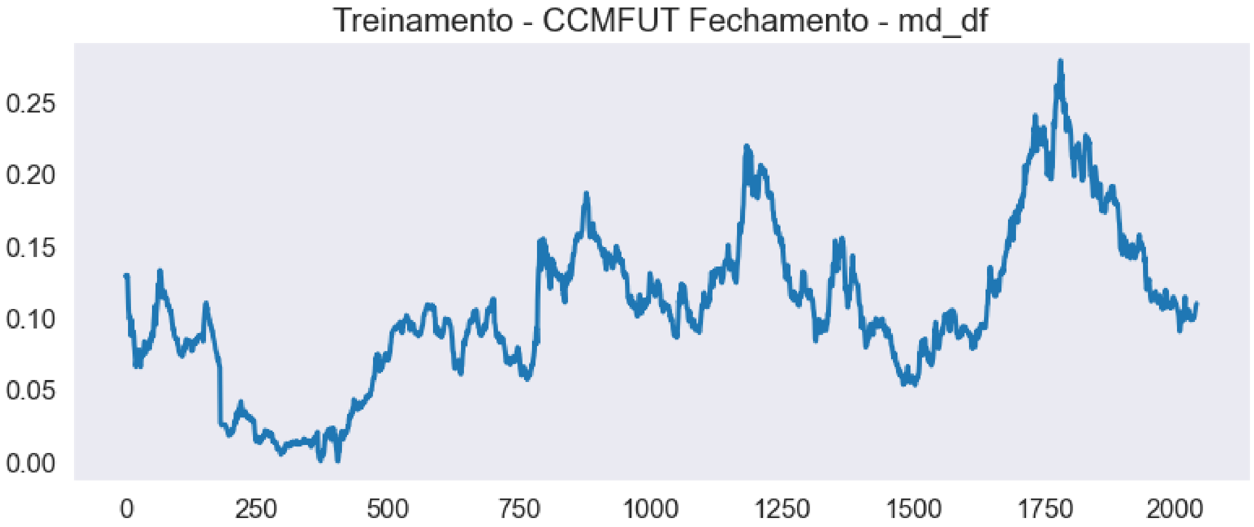
In [325…

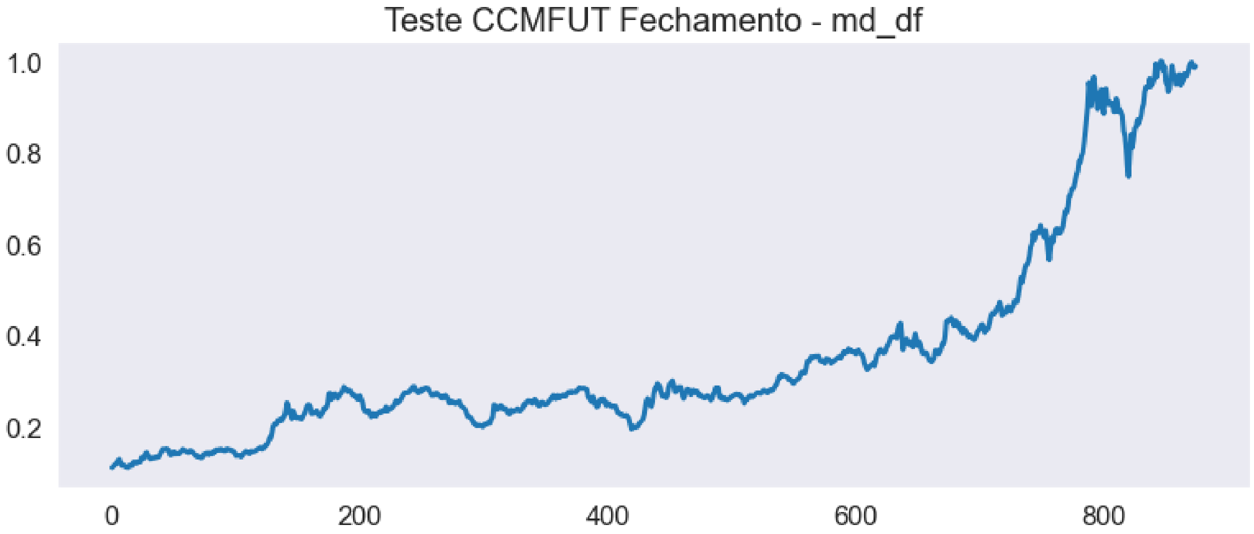
```
split = int(0.70 * len(X))
X_treino = X[:split]
y_treino = y[:split]
```

```
X_teste = X[split:]
y_teste = y[split:]
```

In [326…

```
def show_plot_mc(data, title):
  plt.figure(figsize = (13, 5))
  plt.plot(data, linewidth = 3)
  plt.title(title)
  plt.grid()

show_plot_mc(X_treino, 'Treinamento - CCMFUT Fechamento - md_df')
show_plot_mc(X_teste, 'Teste CCMFUT Fechamento - md_df')
```





In [327…

```
regression_model.fit(X_treino, y_treino)
```

Out[327…

```
Ridge(alpha=1)
```

In [328…

```
lr_accuracy = regression_model.score(X_teste, y_teste)
print("Linear Regression Score: ", lr_accuracy)
```

```
Linear Regression Score:  0.9484580850390733
```

In [329…

```
predicted_prices_mc = regression_model.predict(X)
predicted_prices_mc
```

Out[329…

```
array([[0.12355197],
       [0.12355197],
       [0.12398756],
       ...,
       [0.84466647],
       [0.84292412],
       [0.84553764]])
```

In [330…

```
predicted_mc = []
for i in predicted_prices_mc:
  predicted_mc.append(i[0])
```

In [331…

```
mc_predicao
```

Out[331…

	data	mc_fechamento	mc_fechamento_predito
data			
2008-09-19	2008-09-19	0.128777	0.120601
2008-09-22	2008-09-22	0.128777	0.120452
2008-09-26	2008-09-26	0.129296	0.120939
2008-09-30	2008-09-30	0.127610	0.119349
2008-10-02	2008-10-02	0.128777	0.120427
...	...	...	...
2021-02-24	2021-02-24	0.995461	0.915434
2021-02-25	2021-02-25	0.997536	0.920280
2021-02-26	2021-02-26	0.987550	0.916683
2021-03-01	2021-03-01	0.985475	0.913381
2021-03-02	2021-03-02	0.988588	0.913662

2916 rows × 3 columns

In [332…

```
mc_predicao = mc_predicao.drop(columns='mc_fechamento_predito')
mc_predicao
```

Out[332…

	data	mc_fechamento
data		
2008-09-19	2008-09-19	0.128777
2008-09-22	2008-09-22	0.128777
2008-09-26	2008-09-26	0.129296
2008-09-30	2008-09-30	0.127610
2008-10-02	2008-10-02	0.128777
...	...	...
2021-02-24	2021-02-24	0.995461
2021-02-25	2021-02-25	0.997536

	data	mc_fechamento
data		
2021-02-26	2021-02-26	0.987550
2021-03-01	2021-03-01	0.985475
2021-03-02	2021-03-02	0.988588

2916 rows × 2 columns

In [333...

```
mc_predicao['mc_fechamento_predito'] = predicted_mc
```

In [334...

```
mc_predicao
```

Out[334...

	data	mc_fechamento	mc_fechamento_predito
data			
2008-09-19	2008-09-19	0.128777	0.123552
2008-09-22	2008-09-22	0.128777	0.123552
2008-09-26	2008-09-26	0.129296	0.123988
2008-09-30	2008-09-30	0.127610	0.122572
2008-10-02	2008-10-02	0.128777	0.123552
...	...	...	...
2021-02-24	2021-02-24	0.995461	0.851309
2021-02-25	2021-02-25	0.997536	0.853052
2021-02-26	2021-02-26	0.987550	0.844666
2021-03-01	2021-03-01	0.985475	0.842924
2021-03-02	2021-03-02	0.988588	0.845538

2916 rows × 3 columns

In [335...

```
interactive_plot(mc_predicao, "CCMFUT Fechamento e CCMFUT Fechamento Predito")
```

In [336…

```
#Cálculo do erro
mse = mean_squared_error(mc_predicao['mc_fechamento'], mc_predicao['mc_fechamento_predi
print('MSE: '+str(mse))
mae = mean_absolute_error(mc_predicao['mc_fechamento'], mc_predicao['mc_fechamento_pred
print('MAE: '+str(mae))
rmse = math.sqrt(mean_squared_error(mc_predicao['mc_fechamento'], mc_predicao['mc_fecha
print('RMSE: '+str(rmse))
```

```
MSE: 0.0009740986387099513
MAE: 0.017127976172408
RMSE: 0.03121055332271364
```

In [337…

```
# 5 - RNN - LSTM
```

In [338…

```
mc_df
```

Out[338…

	data	ccmfut_fechamento	ccmfut_fechamento_alvo
data			
2008-09-19	2008-09-19	22.64	22.64
2008-09-22	2008-09-22	22.64	22.68
2008-09-26	2008-09-26	22.68	22.55
2008-09-30	2008-09-30	22.55	22.64
2008-10-02	2008-10-02	22.64	21.65
...	...	...	...
2021-02-24	2021-02-24	89.47	89.63
2021-02-25	2021-02-25	89.63	88.86
2021-02-26	2021-02-26	88.86	88.70
2021-03-01	2021-03-01	88.70	88.94
2021-03-02	2021-03-02	88.94	91.05

2916 rows × 3 columns

In [339…

```
mc_df = mc_df.drop(columns='ccmfut_fechamento_alvo')
mc_df
```

Out[339…

	data	ccmfut_fechamento
data		
2008-09-19	2008-09-19	22.64
2008-09-22	2008-09-22	22.64
2008-09-26	2008-09-26	22.68
2008-09-30	2008-09-30	22.55
2008-10-02	2008-10-02	22.64
...	...	...
2021-02-24	2021-02-24	89.47
2021-02-25	2021-02-25	89.63
2021-02-26	2021-02-26	88.86
2021-03-01	2021-03-01	88.70
2021-03-02	2021-03-02	88.94

2916 rows × 2 columns

In [340…

```
training_data = mc_df.iloc[:, 1:].values
training_data
```

Out[340…

```
array([[22.64],
       [22.64],
       [22.68],
       ...,
       [88.86],
       [88.7 ],
       [88.94]])
```

In [341…

```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(training_data)
```

In [342…

```
training_set_scaled
```

Out[342…

```
array([[0.12877707],
       [0.12877707],
       [0.12929581],
       ...,
       [0.98755025],
       [0.9854753 ],
       [0.98858773]])
```

In [343…

```
X = []
y = []
for i in range(1, len(mc_df)):
    X.append(training_set_scaled [i-1:i, 0])
    y.append(training_set_scaled [i, 0])
```

```
In [344…  X
```

```
Out[344…  [array([0.12877707]),
           array([0.12877707]),
           array([0.12929581]),
           array([0.12760991]),
           array([0.12877707]),
           array([0.11593827]),
           array([0.10647127]),
           array([0.10076514]),
           array([0.09830113]),
           array([0.10063546]),
           array([0.0874076]),
           array([0.09843081]),
           array([0.09246531]),
           array([0.09013098]),
           array([0.08896382]),
           array([0.09013098]),
           array([0.08727791]),
           array([0.08079367]),
           array([0.07729218]),
           array([0.0719751]),
           array([0.06562054]),
           array([0.06912203]),
           array([0.06912203]),
           array([0.06795487]),
           array([0.0719751]),
           array([0.07262352]),
           array([0.07729218]),
           array([0.07145636]),
           array([0.06678771]),
           array([0.06562054]),
           array([0.0702892]),
           array([0.07612502]),
           array([0.07262352]),
           array([0.07495785]),
           array([0.07392037]),
           array([0.07962651]),
           array([0.083128]),
           array([0.08260926]),
           array([0.07495785]),
           array([0.07962651]),
           array([0.08196084]),
           array([0.08079367]),
           array([0.07975619]),
           array([0.08092336]),
           array([0.08196084]),
           array([0.08079367]),
           array([0.07845934]),
           array([0.0814421]),
           array([0.08611075]),
           array([0.08377642]),
           array([0.0874076]),
           array([0.08974193]),
           array([0.08662949]),
           array([0.08675918]),
           array([0.0959668]),
           array([0.10050577]),
           array([0.10673064]),
           array([0.10828686]),
           array([0.09479964]),
           array([0.10296978]),
           array([0.10776812]),
           array([0.1113993]),
           array([0.12294125]),
           array([0.11593827]),
           array([0.12060693]),
           array([0.12644274]),
           array([0.1327973]),
           array([0.12877707]),
           array([0.12760991]),
           array([0.12177409]),
           array([0.11360394]),
           array([0.11477111]),
           array([0.11710543]),
           array([0.11528985]),
           array([0.1182726]),
           array([0.11593827]),
           array([0.11373363]),
           array([0.11295552]),
           array([0.11243678]),
           array([0.11295552]),
           array([0.10776812]),
           array([0.10776812]),
           array([0.10893529]),
           array([0.10776812]),
           array([0.10530411]),
           array([0.10361821]),
           array([0.10478537]),
           array([0.10180262]),
           array([0.09830113]),
           array([0.0959668]),
           array([0.09363247]),
           array([0.09129815]),
           array([0.09194657]),
           array([0.08779665]),
           array([0.08611075]),
           array([0.08546233]),
           array([0.08546233]),
           array([0.08429516]),
           array([0.08546233]),
           array([0.0814421]),
           array([0.0805343]),
           array([0.07910777]),
           array([0.07612502]),
           array([0.07547659]),
           array([0.07430943]),
           array([0.07495785]),
           array([0.07547659]),
           array([0.07379069]),
           array([0.07275321]),
           array([0.07327195]),
           array([0.07443911]),
           array([0.0771625]),
           array([0.08001556]),
           array([0.07768123]),
           array([0.0788484]),
           array([0.083128]),
           array([0.08468422]),
           array([0.08442485]),
           array([0.08273894]),
           array([0.08131241]),
           array([0.08390611]),
           array([0.08196084]),
           array([0.08325768]),
           array([0.08325768]),
           array([0.08325768]),
           array([0.07832966]),
           array([0.07664376]),
           array([0.07729218]),
           array([0.0771625]),
           array([0.08014525]),
           array([0.08183115]),
           array([0.08390611]),
           array([0.08325768]),
           array([0.08157178]),
           array([0.08234989]),
           array([0.0822202]),
           array([0.08546233]),
           array([0.08442485]),
           array([0.08611075]),
           array([0.08714823]),
           array([0.08611075]),
           array([0.08611075]),
           array([0.08662949]),
           array([0.08766697]),
           array([0.08792634]),
           array([0.08779665]),
           array([0.08662949]),
           array([0.08494359]),
           array([0.08325768]),
           array([0.09389184]),
           array([0.09830113]),
           array([0.1028401]),
           array([0.1071197]),
           array([0.10945403]),
           array([0.10997277]),
           array([0.10556348]),
           array([0.10673064]),
           array([0.10258073]),
           array([0.10167293]),
           array([0.09933861]),
           array([0.09843081]),
           array([0.09505901]),
           array([0.09505901]),
           array([0.0933731]),
           array([0.09220594]),
           array([0.09052004]),
           array([0.09052004]),
           array([0.08675918]),
           array([0.08494359]),
           array([0.08260926]),
           array([0.08157178]),
           array([0.07832966]),
           array([0.07664376]),
           array([0.07534691]),
           array([0.07443911]),
           array([0.07106731]),
           array([0.07041888]),
           array([0.07158605]),
           array([0.06821424]),
           array([0.06639865]),
           array([0.06484243]),
           array([0.02645571]),
           array([0.02593697]),
           array([0.02489949]),
           array([0.02502918]),
           array([0.02502918]),
           array([0.02515886]),
           array([0.02554792]),
           array([0.02528855]),
           array([0.02541823]),
           array([0.02541823]),
           array([0.02425107]),
           array([0.02360265]),
           array([0.02204643]),
           array([0.02152769]),
           array([0.01932305]),
           array([0.01815588]),
           array([0.01776683]),
           array([0.01815588]),
           array([0.01828557]),
           array([0.01841525]),
           array([0.02165737]),
           array([0.02010115]),
           array([0.02074958]),
           array([0.01984178]),
           array([0.02087926]),
           array([0.02178706]),
           array([0.0265854]),
           array([0.02814162]),
           array([0.0256776]),
           array([0.02749319]),
           array([0.03112437]),
           array([0.03397743]),
           array([0.02930878]),
           array([0.03060563]),
           array([0.03112437]),
           array([0.03579302]),
           array([0.03553365]),
           array([0.03864609]),
           array([0.04059136]),
           array([0.04149916]),
           array([0.03864609]),
           array([0.03488523]),
           array([0.03190248]),
           array([0.03358838]),
           array([0.03306964]),
           array([0.03397743]),
           array([0.03397743]),
           array([0.03475554]),
           array([0.03306964]),
           array([0.03112437]),
           array([0.03021657]),
           array([0.03203216]),
           array([0.03164311]),
           array([0.03125405]),
           array([0.03021657]),
           array([0.03060563]),
           array([0.02879004]),
           array([0.02879004]),
           array([0.03060563]),
           array([0.02879004]),
           array([0.02930878]),
           array([0.02879004]),
           array([0.02736351]),
           array([0.0282713]),
           array([0.0282713]),
           array([0.0223058]),
           array([0.01776683]),
           array([0.01556218]),
           array([0.01361691]),
           array([0.01698872]),
           array([0.01634029]),
           array([0.01659966]),
           array([0.0145247]),
           array([0.01595124]),
           array([0.01634029]),
           array([0.01270912]),
           array([0.01361691]),
           array([0.0145247]),
           array([0.01400597]),
           array([0.0154325]),
           array([0.01556218]),
           array([0.01724809]),
           array([0.01646998]),
           array([0.01763714]),
           array([0.01906368]),
           array([0.021398]),
           array([0.02100895]),
           array([0.02061989]),
           array([0.01906368]),
           array([0.02100895]),
           array([0.01906368]),
           array([0.01919336]),
           array([0.01737777]),
           array([0.01932305]),
           array([0.02010115]),
           array([0.01867462]),
           array([0.01919336]),
           array([0.01906368]),
           array([0.01880431]),
           array([0.01595124]),
           array([0.0137466]),
           array([0.01439502]),
           array([0.01309817]),
           array([0.01348723]),
           array([0.01348723]),
           array([0.01296849]),
           array([0.01296849]),
           array([0.01063416]),
           array([0.00829983]),
           array([0.00829983]),
           array([0.00933731]),
           array([0.00920763]),
           array([0.00842952]),
           array([0.00778109]),
           array([0.0059655]),
           array([0.00466866]),
           array([0.00635456]),
           array([0.00609519]),
           array([0.00635456]),
           array([0.00583582]),
           array([0.00752172]),
           array([0.00674361]),
           array([0.00829983]),
           array([0.00972636]),
           array([0.0111529]),
           array([0.0111529]),
           array([0.01206069]),
           array([0.01076384]),
           array([0.0102451]),
           array([0.01154195]),
           array([0.01076384]),
           array([0.01011542]),
           array([0.01206069]),
           array([0.01206069]),
           array([0.01257943]),
           array([0.01089353]),
           array([0.01154195]),
           array([0.01232006]),
           array([0.01348723]),
           array([0.01232006]),
           array([0.01296849]),
           array([0.01296849]),
           array([0.01257943]),
           array([0.01296849]),
           array([0.01219038]),
           array([0.01400597]),
           array([0.01219038]),
           array([0.01257943]),
           array([0.01154195]),
           array([0.01219038]),
           array([0.01219038]),
           array([0.01167164]),
           array([0.01206069]),
           array([0.01257943]),
           array([0.01335754]),
           array([0.01296849]),
           array([0.01257943]),
           array([0.01270912]),
           array([0.01296849]),
           array([0.01556218]),
           array([0.01426534]),
           array([0.01491376]),
           array([0.0145247]),
           array([0.01296849]),
           array([0.01348723]),
           array([0.01348723]),
           array([0.01426534]),
           array([0.01335754]),
           array([0.01348723]),
           array([0.01426534]),
           array([0.01335754]),
           array([0.01141227]),
           array([0.01335754]),
           array([0.00998573]),
           array([0.01141227]),
           array([0.01180132]),
           array([0.01180132]),
           array([0.01348723]),
           array([0.01335754]),
           array([0.01672935]),
           array([0.01478407]),
           array([0.01530281]),
           array([0.01387628]),
           array([0.01387628]),
           array([0.01932305]),
           array([0.02023084]),
           array([0.00661393]),
           array([0.00570613]),
           array([0.00246401]),
           array([0.00116716]),
           array([0.00077811]),
           array([0.00012968]),
           array([0.00233433]),
           array([0.00544676]),
           array([0.0068733]),
           array([0.0059655]),
           array([0.0042796]),
           array([0.00583582]),
           array([0.00985605]),
           array([0.0145247]),
           array([0.01828557]),
           array([0.01517313]),
           array([0.01789651]),
           array([0.01880431]),
           array([0.02100895]),
           array([0.02191674]),
           array([0.021398]),
           array([0.01945273]),
           array([0.02178706]),
           array([0.02243548]),
           array([0.02269485]),
           array([0.02126832]),
           array([0.0145247]),
           array([0.01906368]),
           array([0.02256517]),
           array([0.02074958]),
           array([0.02269485]),
           array([0.01945273]),
           array([0.01867462]),
           array([0.01556218]),
           array([0.0111529]),
           array([0.00868889]),
           array([0.00557645]),
           array([0.00025937]),
           array([0.]),
           array([0.00389055]),
           array([0.00674361]),
           array([0.00700298]),
           array([0.01556218]),
           array([0.01841525]),
           array([0.01893399]),
           array([0.02087926]),
           array([0.02087926]),
           array([0.01517313]),
           array([0.01517313]),
           array([0.01932305]),
           array([0.01789651]),
           array([0.01867462]),
           array([0.02010115]),
           array([0.02074958]),
           array([0.01958241]),
           array([0.02049021]),
           array([0.02269485]),
           array([0.02438075]),
           array([0.02438075]),
           array([0.02930878]),
           array([0.03151342]),
           array([0.02956815]),
           array([0.03216185]),
           array([0.03190248]),
           array([0.03358838]),
           array([0.03592271]),
           array([0.03332901]),
           array([0.03890546]),
           array([0.04292569]),
           array([0.0420179]),
           array([0.03851641]),
           array([0.03734924]),
           array([0.03994294]),
           array([0.03696019]),
           array([0.03696019]),
           array([0.03579302]),
           array([0.03657113]),
           array([0.03916483]),
           array([0.04072105]),
           array([0.0394242]),
           array([0.03696019]),
           array([0.03812735]),
           array([0.0377383]),
           array([0.03825704]),
           array([0.03851641]),
           array([0.04033199]),
           array([0.0411101]),
           array([0.04046168]),
           array([0.04123979]),
           array([0.04072105]),
           array([0.04253664]),
           array([0.04487096]),
           array([0.04318506]),
           array([0.04331475]),
           array([0.04409285]),
           array([0.04551939]),
           array([0.04487096]),
           array([0.04590844]),
           array([0.04668655]),
           array([0.04551939]),
           array([0.04707561]),
           array([0.04953962]),
           array([0.04953962]),
           array([0.05304111]),
           array([0.05550512]),
           array([0.05783945]),
           array([0.0574504]),
           array([0.05900661]),
           array([0.06185968]),
           array([0.06613928]),
           array([0.07158605]),
           array([0.0702892]),
           array([0.07405006]),
           array([0.07314226]),
           array([0.07443911]),
           array([0.07275321]),
           array([0.06626897]),
           array([0.06250811]),
           array([0.06380495]),
           array([0.0634159]),
           array([0.0676955]),
           array([0.06743613]),
           array([0.0651018]),
           array([0.06743613]),
           array([0.06873298]),
           array([0.07119699]),
           array([0.07301258]),
           array([0.07275321]),
           array([0.07469848]),
           array([0.07495785]),
           array([0.07119699]),
           array([0.07301258]),
           array([0.0719751]),
           array([0.07015951]),
           array([0.07067825]),
           array([0.07262352]),
           array([0.07417974]),
           array([0.07573596]),
           array([0.0805343]),
           array([0.08273894]),
           array([0.08831539]),
           array([0.08792634]),
           array([0.09077941]),
           array([0.09103878]),
           array([0.09142783]),
           array([0.09233562]),
           array([0.09298405]),
           array([0.09220594]),
           array([0.09272468]),
           array([0.09181688]),
           array([0.09272468]),
           array([0.09479964]),
           array([0.09492932]),
           array([0.09402153]),
           array([0.09311373]),
           array([0.09531838]),
           array([0.09674491]),
           array([0.09583712]),
           array([0.09518869]),
           array([0.09454027]),
           array([0.08922319]),
           array([0.09492932]),
           array([0.09402153]),
           array([0.09544806]),
           array([0.09778239]),
           array([0.09804176]),
           array([0.09778239]),
           array([0.09869018]),
           array([0.10128388]),
           array([0.10076514]),
           array([0.10037609]),
           array([0.09752302]),
           array([0.0976527]),
           array([0.09661523]),
           array([0.09583712]),
           array([0.09103878]),
           array([0.09298405]),
           array([0.09661523]),
           array([0.09544806]),
           array([0.09181688]),
           array([0.09350279]),
           array([0.09544806]),
           array([0.09272468]),
           array([0.09181688]),
           array([0.09013098]),
           array([0.08883413]),
           array([0.08870445]),
           array([0.08948256]),
           array([0.08922319]),
           array([0.0890935]),
           array([0.08779665]),
           array([0.08688886]),
           array([0.08805602]),
           array([0.0890935]),
           array([0.08831539]),
           array([0.08831539]),
           array([0.0900013]),
           array([0.09220594]),
           array([0.09350279]),
           array([0.09415121]),
           array([0.09713396]),
           array([0.1011542]),
           array([0.1019323]),
           array([0.10206199]),
           array([0.104526]),
           array([0.10504474]),
           array([0.10634159]),
           array([0.10763844]),
           array([0.10893529]),
           array([0.10802749]),
           array([0.10802749]),
           array([0.10802749]),
           array([0.10634159]),
           array([0.10582285]),
           array([0.10595254]),
           array([0.10595254]),
           array([0.10647127]),
           array([0.10867592]),
           array([0.10595254]),
           array([0.10673064]),
           array([0.10595254]),
           array([0.10673064]),
           array([0.10089483]),
           array([0.09946829]),
           array([0.09026067]),
           array([0.0890935]),
           array([0.0900013]),
           array([0.0933731]),
           array([0.09116846]),
           array([0.08779665]),
           array([0.08779665]),
           array([0.09116846]),
           array([0.09116846]),
           array([0.09026067]),
           array([0.08870445]),
           array([0.08611075]),
           array([0.08779665]),
           array([0.08831539]),
           array([0.08870445]),
           array([0.09077941]),
           array([0.09415121]),
           array([0.09674491]),
           array([0.09920892]),
           array([0.09881987]),
           array([0.09700428]),
           array([0.09791207]),
           array([0.0985605]),
           array([0.09739333]),
           array([0.09830113]),
           array([0.09830113]),
           array([0.09713396]),
           array([0.09466995]),
           array([0.09454027]),
           array([0.09285436]),
           array([0.0890935]),
           array([0.08494359]),
           array([0.07858903]),
           array([0.07560628]),
           array([0.07521722]),
           array([0.06847361]),
           array([0.06471275]),
           array([0.06432369]),
           array([0.06536117]),
           array([0.06562054]),
           array([0.06743613]),
           array([0.06756582]),
           array([0.06847361]),
           array([0.07015951]),
           array([0.06912203]),
           array([0.06899235]),
           array([0.06484243]),
           array([0.06147063]),
           array([0.06056283]),
           array([0.06484243]),
           array([0.06626897]),
           array([0.07301258]),
           array([0.07677344]),
           array([0.08286863]),
           array([0.08364674]),
           array([0.08066399]),
           array([0.08364674]),
           array([0.08533264]),
           array([0.08727791]),
           array([0.08779665]),
           array([0.09077941]),
           array([0.08948256]),
           array([0.09039035]),
           array([0.09376216]),
           array([0.09674491]),
           array([0.09752302]),
           array([0.09830113]),
           array([0.09817144]),
           array([0.09674491]),
           array([0.09661523]),
           array([0.09791207]),
           array([0.09985735]),
           array([0.10258073]),
           array([0.10439632]),
           array([0.09869018]),
           array([0.10011672]),
           array([0.10387758]),
           array([0.10517443]),
           array([0.10439632]),
           array([0.09972766]),
           array([0.09622617]),
           array([0.09389184]),
           array([0.09389184]),
           array([0.09064972]),
           array([0.08883413]),
           array([0.09090909]),
           array([0.08727791]),
           array([0.08727791]),
           array([0.08870445]),
           array([0.08961224]),
           array([0.08922319]),
           array([0.09090909]),
           array([0.09246531]),
           array([0.09518869]),
           array([0.0933731]),
           array([0.09233562]),
           array([0.09518869]),
           array([0.09817144]),
           array([0.10154325]),
           array([0.10504474]),
           array([0.10582285]),
           array([0.10647127]),
           array([0.10387758]),
           array([0.10387758]),
           array([0.10439632]),
           array([0.10673064]),
           array([0.10906497]),
           array([0.11101025]),
           array([0.11126961]),
           array([0.11282583]),
           array([0.11282583]),
           array([0.11243678]),
           array([0.09635586]),
           array([0.09531838]),
           array([0.09207625]),
           array([0.08701854]),
           array([0.08688886]),
           array([0.08598107]),
           array([0.08779665]),
           array([0.08883413]),
           array([0.08675918]),
           array([0.08520296]),
           array([0.08247957]),
           array([0.08585138]),
           array([0.08624044]),
           array([0.08507327]),
           array([0.08390611]),
           array([0.08429516]),
           array([0.08559201]),
           array([0.08351705]),
           array([0.08157178]),
           array([0.07923745]),
           array([0.07690313]),
           array([0.07392037]),
           array([0.06977046]),
           array([0.06860329]),
           array([0.07002983]),
           array([0.06977046]),
           array([0.07015951]),
           array([0.07210479]),
           array([0.06990014]),
           array([0.06782518]),
           array([0.06704708]),
           array([0.06977046]),
           array([0.06834392]),
           array([0.07132668]),
           array([0.07080794]),
           array([0.07119699]),
           array([0.07262352]),
           array([0.07158605]),
           array([0.07015951]),
           array([0.06925172]),
           array([0.07067825]),
           array([0.07210479]),
           array([0.07586565]),
           array([0.07781092]),
           array([0.07910777]),
           array([0.07871871]),
           array([0.07586565]),
           array([0.07443911]),
           array([0.07093762]),
           array([0.06549086]),
           array([0.06263779]),
           array([0.05991441]),
           array([0.05978472]),
           array([0.06121126]),
           array([0.06069252]),
           array([0.06549086]),
           array([0.06575023]),
           array([0.06432369]),
           array([0.06354558]),
           array([0.06328621]),
           array([0.06237842]),
           array([0.05758008]),
           array([0.05887693]),
           array([0.05706134]),
           array([0.05783945]),
           array([0.05965504]),
           array([0.05965504]),
           array([0.05978472]),
           array([0.05978472]),
           array([0.06263779]),
           array([0.06445338]),
           array([0.06730645]),
           array([0.06626897]),
           array([0.06730645]),
           array([0.07093762]),
           array([0.07327195]),
           array([0.07664376]),
           array([0.08247957]),
           array([0.08105304]),
           array([0.08922319]),
           array([0.09259499]),
           array([0.08196084]),
           array([0.11191804]),
           array([0.12527558]),
           array([0.13396447]),
           array([0.13552068]),
           array([0.144858]),
           array([0.15328751]),
           array([0.1491376]),
           array([0.13305667]),
           array([0.14680327]),
           array([0.13980029]),
           array([0.14005966]),
           array([0.15471404]),
           array([0.15004539]),
           array([0.1456361]),
           array([0.14784075]),
           array([0.15004539]),
           array([0.14511736]),
           array([0.13837375]),
           array([0.13487226]),
           array([0.14369083]),
           array([0.13980029]),
           array([0.12903644]),
           array([0.12644274]),
           array([0.12281157]),
           array([0.12008819]),
           array([0.12358968]),
           array([0.13214888]),
           array([0.14070808]),
           array([0.13837375]),
           array([0.13668785]),
           array([0.13889249]),
           array([0.13526132]),
           array([0.13461289]),
           array([0.13396447]),
           array([0.13344573]),
           array([0.13059266]),
           array([0.1285177]),
           array([0.13124108]),
           array([0.12877707]),
           array([0.1294255]),
           array([0.12812865]),
           array([0.12657243]),
           array([0.12760991]),
           array([0.12618337]),
           array([0.12670211]),
           array([0.12631306]),
           array([0.12929581]),
           array([0.12008819]),
           array([0.12073661]),
           array([0.11516016]),
           array([0.11840228]),
           array([0.11334457]),
           array([0.11049151]),
           array([0.12462716]),
           array([0.12786928]),
           array([0.12242251]),
           array([0.12670211]),
           array([0.13072234]),
           array([0.12929581]),
           array([0.12618337]),
           array([0.12890676]),
           array([0.12605369]),
           array([0.13616911]),
           array([0.12838802]),
           array([0.1294255]),
           array([0.13188951]),
           array([0.13603942]),
           array([0.13902218]),
           array([0.14382052]),
           array([0.14382052]),
           array([0.14472831]),
           array([0.14939697]),
           array([0.15147192]),
           array([0.15471404]),
           array([0.15289846]),
           array([0.15536247]),
           array([0.15782648]),
           array([0.156789]),
           array([0.15717806]),
           array([0.1576968]),
           array([0.156789]),
           array([0.156789]),
           array([0.15549215]),
           array([0.15665932]),
           array([0.15925302]),
           array([0.16029049]),
           array([0.16599663]),
           array([0.17351835]),
           array([0.17701984]),
           array([0.17701984]),
           array([0.1765011]),
           array([0.17753858]),
           array([0.18129944]),
           array([0.18648684]),
           array([0.18259629]),
           array([0.18324472]),
           array([0.17935417]),
           array([0.1765011]),
           array([0.16405136]),
           array([0.15886396]),
           array([0.15562184]),
           array([0.15795617]),
           array([0.16184671]),
           array([0.16132797]),
           array([0.16314356]),
           array([0.1653482]),
           array([0.1653482]),
           array([0.15977175]),
           array([0.1551031]),
           array([0.15743743]),
           array([0.15795617]),
           array([0.15367657]),
           array([0.15393594]),
           array([0.15601089]),
           array([0.15588121]),
           array([0.15380625]),
           array([0.15406562]),
           array([0.15406562]),
           array([0.15030476]),
           array([0.14835949]),
           array([0.14835949]),
           array([0.14745169]),
           array([0.14758138]),
           array([0.14446894]),
           array([0.14278304]),
           array([0.14628453]),
           array([0.14693295]),
           array([0.14706264]),
           array([0.144858]),
           array([0.13850344]),
           array([0.135391]),
           array([0.13318636]),
           array([0.13487226]),
           array([0.13681753]),
           array([0.1413565]),
           array([0.14459863]),
           array([0.14096745]),
           array([0.13980029]),
           array([0.14174556]),
           array([0.14174556]),
           array([0.14109713]),
           array([0.14096745]),
           array([0.13941123]),
           array([0.13422384]),
           array([0.13461289]),
           array([0.13759564]),
           array([0.13837375]),
           array([0.14161587]),
           array([0.14382052]),
           array([0.14719232]),
           array([0.14952665]),
           array([0.14719232]),
           array([0.14382052]),
           array([0.14304241]),
           array([0.14550642]),
           array([0.14096745]),
           array([0.13837375]),
           array([0.13915186]),
           array([0.13992997]),
           array([0.14044871]),
           array([0.13733627]),
           array([0.1396706]),
           array([0.13474258]),
           array([0.13098171]),
           array([0.13150045]),
           array([0.12968487]),
           array([0.12773959]),
           array([0.12825833]),
           array([0.13033329]),
           array([0.12981455]),
           array([0.12760991]),
           array([0.12203346]),
           array([0.11334457]),
           array([0.11386331]),
           array([0.11412268]),
           array([0.11062119]),
           array([0.10958371]),
           array([0.11347426]),
           array([0.11477111]),
           array([0.10997277]),
           array([0.10789781]),
           array([0.10841655]),
           array([0.10673064]),
           array([0.11049151]),
           array([0.10984308]),
           array([0.10750875]),
           array([0.10660096]),
           array([0.10595254]),
           array([0.10361821]),
           array([0.10180262]),
           array([0.10063546]),
           array([0.10128388]),
           array([0.10608222]),
           array([0.10945403]),
           array([0.11554922]),
           array([0.1156789]),
           array([0.10374789]),
           array([0.10517443]),
           array([0.1028401]),
           array([0.1088056]),
           array([0.10556348]),
           array([0.11049151]),
           array([0.11204772]),
           array([0.10958371]),
           array([0.10660096]),
           array([0.10997277]),
           array([0.11178835]),
           array([0.10815718]),
           array([0.11113993]),
           array([0.11178835]),
           array([0.11295552]),
           array([0.11049151]),
           array([0.11749449]),
           array([0.12449747]),
           array([0.13059266]),
           ...]
```

In [345…

```
y
```

Out[345…

```
[0.1287770717157308,
 0.12929581117883546,
 0.12760990792374535,
 0.1287770717157308,
 0.11593827000389054,
 0.10647127480223059,
 0.10076514070807938,
 0.09830112825833223,
 0.10063545584230318,
 0.08740759953313451,
 0.09843081312410842,
 0.09246530929840488,
 0.09013098171443393,
 0.08896381792244848,
 0.09013098171443393,
 0.08727791466735832,
 0.08079367137855015,
 0.07729218000259372,
 0.071975100505771,
 0.06562054208273893,
 0.06912203345869536,
 0.06912203345869536,
 0.06795486966670988,
 0.071975100505771,
 0.07262352483465179,
 0.07729218000259372,
 0.07145636104266631,
 0.0667877058747244,
 0.06562054208273893,
 0.07028919725068083,
 0.07612501621060822,
 0.07262352483465179,
 0.07495785241862274,
 0.07392037349241345,
 0.07962650758656467,
 0.0831279989625211,
 0.08260925949941642,
 0.07495785241862274,
 0.07962650758656467,
 0.08196083517053562,
 0.08079367137855015,
 0.07975619245234081,
 0.08092335624432628,
 0.08196083517053562,
 0.08079367137855015,
 0.0784593437945792,
 0.08144209570743094,
 0.08611075087537287,
 0.08377642329140192,
 0.08740759953313451,
 0.08974192711710541,
 0.08662949033847753,
 0.08675917520425366,
 0.09596680067436128,
 0.10050577097652705,
 0.10673064453378292,
 0.10828686292309689,
 0.09479963688237583,
 0.10296978342627414,
 0.10776812345999223,
 0.11139929970172482,
 0.12294125275580345,
 0.11593827000389054,
 0.1206069251718325,
 0.12644274413175985,
 0.13279730255479186,
 0.1287770717157308,
 0.12760990792374535,
 0.12177408896381794,
 0.11360394241991958,
 0.11477110621190509,
 0.11710543379587604,
 0.11528984567500974,
 0.11827259758786149,
 0.11593827000389054,
 0.11373362728569578,
 0.11295551809103879,
 0.11243677862793414,
 0.11295551809103879,
 0.10776812345999223,
 0.10776812345999223,
 0.10893528725197768,
 0.10776812345999223,
 0.10530411101024509,
 0.10361820775515498,
 0.10478537154714043,
 0.10180261963428869,
 0.09830112825833223,
 0.09596680067436128,
 0.09363247309039038,
 0.09129814550641943,
 0.09194656983530022,
 0.08779665413046298,
 0.08611075087537287,
 0.08546232654649202,
 0.08546232654649202,
 0.08429516275450658,
 0.08546232654649202,
 0.08144209570743094,
 0.08053430164699782,
 0.07910776812345999,
 0.07612501621060822,
 0.07547659188172742,
 0.07430942808974195,
 0.07495785241862274,
 0.07547659188172742,
 0.07379068862663726,
 0.07275320970042798,
 0.07327194916353261,
 0.07443911295551808,
 0.07716249513681755,
 0.08001556218389314,
 0.07768123459992218,
 0.07884839839190766,
 0.0831279989625211,
 0.08468421735183504,
 0.08442484762028271,
 0.08273894436519258,
 0.0813124108416548,
 0.08390610815717806,
 0.08196083517053562,
 0.08325768382829724,
 0.08325768382829724,
 0.08325768382829724,
 0.078329658928803,
 0.0766437556737129,
 0.07729218000259372,
 0.07716249513681755,
 0.08014524704966933,
 0.08183115030475943,
 0.08390610815717806,
 0.08325768382829724,
 0.0815717805732071,
 0.08234988976786409,
 0.08222020490208795,
 0.08546232654649202,
 0.08442484762028271,
 0.08611075087537287,
 0.08714822980158218,
 0.08611075087537287,
 0.08611075087537287,
 0.08662949033847753,
 0.08766696926468678,
 0.08792633899623911,
 0.08779665413046298,
 0.08662949033847753,
 0.0849435870833874,
 0.08325768382829724,
 0.09389184282194266,
 0.09830112825833223,
 0.102840098560498,
 0.10711969913111138,
 0.10945402671508234,
 0.10997276617818705,
 0.10556348074179747,
 0.10673064453378292,
 0.10258072882894567,
 0.10167293476851255,
 0.0993386071845416,
 0.09843081312410842,
 0.09505900661392816,
 0.09505900661392816,
 0.09337310335883806,
 0.09220593956685255,
 0.09052003631176245,
 0.09052003631176245,
 0.08675917520425366,
 0.0849435870833874,
 0.08260925949941642,
 0.0815717805732071,
 0.078329658928803,
 0.0766437556737129,
 0.07534690701595123,
 0.07443911295551808,
 0.07106730644533785,
 0.07041888211645703,
 0.0715860459084425,
 0.06821423939826221,
 0.06639865127739591,
 0.06484243288808197,
 0.02645571261833743,
 0.0259369731552328,
 0.02489949422902349,
 0.025029179094799653,
 0.025029179094799653,
 0.02515886396057579,
 0.02554791855790428,
 0.025288548826351953,
 0.025418233692128117,
 0.025418233692128117,
 0.02425106990014264,
 0.02360264557126182,
 0.022046427181947853,
 0.021527687718843197,
 0.01932304500064841,
 0.018155881208662933,
 0.01776682661133444,
 0.018155881208662933,
 0.018285566074439097,
 0.01841525094021529,
 0.02165737258461939,
 0.020101154195305393,
 0.02074957852418624,
 0.019841784463753065,
 0.020879263389962377,
 0.021787057450395553,
 0.026585397484113593,
 0.02814161587342759,
 0.025677603423680445,
 0.02749319154454674,
 0.031124367786279333,
 0.03397743483335494,
 0.029308779665413037,
 0.030605628323174677,
 0.031124367786279333,
 0.03579302295422124,
 0.03553365322266891,
 0.038646090001296846,
 0.040591362987939306,
 0.041499157048372454,
 0.038646090001296846,
 0.03488522889378809,
 0.03190247698093632,
 0.03358838023602645,
 0.03306964077292179,
 0.03397743483335494,
 0.03397743483335494,
 0.034755544028011925,
 0.03306964077292179,
 0.031124367786279333,
 0.030216573725846185,
 0.03203216184671248,
 0.03164310724938399,
 0.0312540526520555,
 0.030216573725846185,
 0.030605628323174677,
 0.02879004020230838,
 0.02879004020230838,
 0.030605628323174677,
 0.02879004020230838,
 0.029308779665413037,
 0.02879004020230838,
 0.027363506678770577,
 0.028271300739203725,
 0.028271300739203725,
 0.02230579691350018,
 0.01776682661133444,
 0.015562183893139653,
 0.013616910906497193,
 0.016988717416677457,
 0.016340293087796665,
 0.016599662819348965,
 0.01452470496693034,
 0.015951238490468145,
 0.016340293087796665,
 0.012709116846064045,
 0.013616910906497193,
 0.01452470496693034,
 0.014005965503825685,
 0.015432499027363517,
 0.015562183893139653,
 0.017248087148229785,
 0.01646997795357283,
 0.017637141745558277,
 0.01906367526909608,
 0.021398002853067033,
 0.02100894825573854,
 0.020619893658410077,
 0.01906367526909608,
 0.02100894825573854,
 0.01906367526909608,
 0.019193360134872245,
 0.017377772014005977,
 0.01932304500064841,
 0.020101154195305393,
 0.018674620671767617,
 0.019193360134872245,
 0.01906367526909608,
 0.018804305537543753,
 0.015951238490468145,
 0.013746595772273357,
 0.014395020101154204,
 0.013098171443392564,
 0.013487226040721029,
 0.013487226040721029,
 0.0129684865776164,
 0.0129684865776164,
 0.01063415899364542,
 0.008299831409674469,
 0.008299831409674469,
 0.00933731033588378,
 0.009207625470107617,
 0.008429516275450633,
 0.00778109194656984,
 0.0059655038257035164,
 0.004668655167941904,
 0.0063545584230320085,
 0.0060951886914796805,
 0.0063545584230320085,
 0.00583581895992738,
 0.0075217222150174845,
 0.006743613020360528,
 0.008299831409674469,
 0.0097263649332123,
 0.011152898456750104,
 0.011152898456750104,
 0.012060692517183252,
 0.010763843859421585,
 0.010245104396316929,
 0.011541953054078569,
 0.010763843859421585,
 0.010115419530540792,
 0.012060692517183252,
 0.012060692517183252,
 0.01257943198028788,
 0.010893528725197776,
 0.011541953054078569,
 0.01232006224873558,
 0.013487226040721029,
 0.01232006224873558,
 0.0129684865776164,
 0.0129684865776164,
 0.01257943198028788,
 0.0129684865776164,
 0.012190377382959416,
 0.014005965503825685,
 0.012190377382959416,
 0.01257943198028788,
 0.011541953054078569,
 0.012190377382959416,
 0.012190377382959416,
 0.011671637919854733,
 0.012060692517183252,
 0.01257943198028788,
 0.013357541174944892,
 0.0129684865776164,
 0.01257943198028788,
 0.012709116846064045,
 0.0129684865776164,
 0.015562183893139653,
 0.01426533523537804,
 0.014913759564258833,
 0.01452470496693034,
 0.0129684865776164,
 0.013487226040721029,
 0.013487226040721029,
 0.01426533523537804,
 0.013357541174944892,
 0.013487226040721029,
 0.01426533523537804,
 0.013357541174944892,
 0.011412268188302405,
 0.013357541174944892,
 0.009985734664764628,
 0.011412268188302405,
 0.011801322785630897,
 0.011801322785630897,
 0.013487226040721029,
 0.013357541174944892,
 0.01672934768512513,
 0.014784074698482669,
 0.015302814161587353,
 0.01387628063804952,
 0.01387628063804952,
 0.01932304500064841,
 0.020230839061081557,
 0.006613928154584364,
 0.005706134094151216,
 0.002464012449747116,
 0.001167163791985476,
 0.0007781091946569563,
 0.000129684865776164,
 0.002334327583970952,
 0.005446764362598888,
 0.006873297886136692,
 0.0059655038257035164,
 0.004279600570613384,
 0.00583581895992738,
 0.009856049798988464,
 0.01452470496693034,
 0.018285566074439097,
 0.015173129295811189,
 0.017896511477110605,
 0.018804305537543753,
 0.02100894825573854,
 0.02191674231617169,
 0.021398002853067033,
 0.0194527298664246,
 0.021787057450395553,
 0.022435481779276345,
 0.0226948515108287,
 0.02126831798729087,
 0.01452470496693034,
 0.01906367526909608,
 0.02256516664505251,
 0.02074957852418624,
 0.0226948515108287,
 0.0194527298664246,
 0.018674620671767617,
 0.015562183893139653,
 0.011152898456750104,
 0.008688886007002988,
 0.005576449228375052,
 0.000259369731552328,
 0.0,
 0.0038905459732848924,
 0.006743613020360528,
 0.007002982751912856,
 0.015562183893139653,
 0.01841525094021529,
 0.018933990403319917,
 0.020879263389962377,
 0.020879263389962377,
 0.015173129295811189,
 0.015173129295811189,
 0.01932304500064841,
 0.017896511477110605,
 0.018674620671767617,
 0.020101154195305393,
 0.02074957852418624,
 0.019582414732200765,
 0.020490208792633885,
 0.0226948515108287,
 0.024380754765918805,
 0.024380754765918805,
 0.029308779665413037,
 0.031513422383607825,
 0.029568149396965365,
 0.032161846712488645,
 0.03190247698093632,
 0.03358838023602645,
 0.0359227078199974,
 0.03332901050447412,
 0.038905459732849174,
 0.04292569057191026,
 0.04201789651147711,
 0.03851640513552068,
 0.037349241343535206,
 0.039942938659058486,
 0.036960186746206714,
 0.036960186746206714,
 0.03579302295422124,
 0.03657113214887822,
 0.0391648294644015,
 0.04072104785371547,
 0.03942419919595383,
 0.036960186746206714,
 0.03812735053819219,
 0.0377382959408637,
 0.038257035403968354,
 0.03851640513552068,
 0.04033199325638698,
 0.04111010245104396,
 0.04046167812216314,
 0.041239787316820126,
 0.04072104785371547,
 0.042536635974581766,
 0.044870963558552746,
 0.043185060303462586,
 0.04331474516923875,
 0.044092854363895734,
 0.04551938788743351,
 0.044870963558552746,
 0.04590844248476203,
 0.046686551679418986,
 0.04551938788743351,
 0.047075606276747506,
 0.04953961872649462,
 0.04953961872649462,
 0.05304111010245105,
 0.05550512255219814,
 0.057839450136169146,
 0.057450395538840626,
 0.059006613928154594,
 0.0618596809752302,
 0.06613928154584359,
 0.0715860459084425,
 0.07028919725068083,
 0.07405005835818962,
 0.07314226429775647,
 0.07443911295551808,
 0.07275320970042798,
 0.06626896641161978,
 0.06250810530411102,
 0.06380495396187263,
 0.06341589936454417,
 0.06769549993515758,
 0.06743613020360525,
 0.0651018026196343,
 0.06743613020360525,
 0.0687329788613669,
 0.07119699131111398,
 0.07301257943198028,
 0.07275320970042798,
 0.07469848268707041,
 0.07495785241862274,
 0.07119699131111398,
 0.07301257943198028,
 0.071975100505771,
 0.0701595123849047,
 0.07067825184800935,
 0.07262352483465179,
 0.07417974322396576,
 0.07573596161327975,
 0.08053430164699782,
 0.08273894436519258,
 0.08831539359356763,
 0.08792633899623911,
 0.09077940604331478,
 0.0910387757748671,
 0.09142783037219562,
 0.09233562443262869,
 0.09298404876150954,
 0.09220593956685255,
 0.09272467902995721,
 0.09181688496952403,
 0.09272467902995721,
 0.09479963688237583,
 0.09492932174815202,
 0.09402152768771885,
 0.09311373362728573,
 0.09531837634548049,
 0.09674490986901832,
 0.09583711580858514,
 0.09518869147970435,
 0.0945402671508235,
 0.08922318765400081,
 0.09492932174815202,
 0.09402152768771885,
 0.09544806121125668,
 0.09778238879522763,
 0.09804175852677996,
 0.09778238879522763,
 0.09869018285566075,
 0.10128388017118403,
 0.10076514070807938,
 0.10037608611075086,
 0.0975230190636753,
 0.09765270392945144,
 0.09661522500324213,
 0.09583711580858514,
 0.0910387757748671,
 0.09298404876150954,
 0.09661522500324213,
 0.09544806121125668,
 0.09181688496952403,
 0.09350278822461425,
 0.09544806121125668,
 0.09272467902995721,
 0.09181688496952403,
 0.09013098171443393,
 0.08883413305667229,
 0.08870444819089615,
 0.08948255738555314,
 0.08922318765400081,
 0.08909350278822462,
 0.08779665413046298,
 0.08688886007002986,
 0.0880560238620153,
 0.08909350278822462,
 0.08831539359356763,
 0.08831539359356763,
 0.09000129684865774,
 0.09220593956685255,
 0.09350278822461425,
 0.09415121255349498,
 0.09713396446634678,
 0.1011541953054079,
 0.10193230450006488,
 0.10206198936584102,
 0.1045260018155881,
 0.10504474127869276,
 0.10634158993645446,
 0.1076384385942161,
 0.10893528725197768,
 0.10802749319154456,
 0.10802749319154456,
 0.10802749319154456,
 0.10634158993645446,
 0.1058228504733498,
 0.10595253533912594,
 0.10595253533912594,
 0.10647127480223059,
 0.10867591752042535,
 0.10595253533912594,
 0.10673064453378292,
 0.10595253533912594,
 0.10673064453378292,
 0.10089482557385551,
 0.09946829205031774,
 0.09026066658021012,
 0.08909350278822462,
 0.09000129684865774,
 0.09337310335883806,
 0.09116846064064324,
 0.08779665413046298,
 0.08779665413046298,
 0.09116846064064324,
 0.09116846064064324,
 0.09026066658021012,
 0.08870444819089615,
 0.08611075087537287,
 0.08779665413046298,
 0.08831539359356763,
 0.08870444819089615,
 0.09077940604331478,
 0.09415121255349498,
 0.09674490986901832,
 0.09920892231876541,
 0.09881986772143689,
 0.09700427960057065,
 0.09791207366100377,
 0.09856049798988456,
 0.09739333419789911,
 0.09830112825833223,
 0.09830112825833223,
 0.09713396446634678,
 0.0946699520165997,
 0.0945402671508235,
 0.0928543638957334,
 0.08909350278822462,
 0.0849435870833874,
 0.07858902866035533,
 0.07560627674750356,
 0.0752172221501751,
 0.06847360912981454,
 0.06471274802230578,
 0.06432369342497732,
 0.06536117235118663,
 0.06562054208273893,
 0.06743613020360525,
 0.06756581506938142,
 0.06847360912981454,
 0.0701595123849047,
 0.06912203345869536,
 0.06899234859291922,
 0.06484243288808197,
 0.06147062637790168,
 0.060562832317468535,
 0.06484243288808197,
 0.06626896641161978,
 0.07301257943198028,
 0.07677344053948904,
 0.0828686292309687,
 0.08364673842562573,
 0.08066398651277396,
 0.08364673842562573,
 0.08533264168071583,
 0.08727791466735832,
 0.08779665413046298,
 0.09077940604331478,
 0.08948255738555314,
 0.09039035144598626,
 0.09376215795616652,
 0.09674490986901832,
 0.0975230190636753,
 0.09830112825833223,
 0.0981714433925561,
 0.09674490986901832,
 0.09661522500324213,
 0.09791207366100377,
 0.09985734664764626,
 0.10258072882894567,
 0.10439631694981197,
 0.09869018285566075,
 0.10011671637919858,
 0.10387757748670731,
 0.10517442614446895,
 0.10439631694981197,
 0.09972766178187006,
 0.09622617040591361,
 0.09389184282194266,
 0.09389184282194266,
 0.09064972117753858,
 0.08883413305667229,
 0.09090909090909091,
 0.08727791466735832,
 0.08727791466735832,
 0.08870444819089615,
 0.08961224225132927,
 0.08922318765400081,
 0.09090909090909091,
 0.09246530929840488,
 0.09518869147970435,
 0.09337310335883806,
 0.09233562443262869,
 0.09518869147970435,
 0.0981714433925561,
 0.10154324990273636,
 0.10504474127869276,
 0.1058228504733498,
 0.10647127480223059,
 0.10387757748670731,
 0.10387757748670731,
 0.10439631694981197,
 0.10673064453378292,
 0.10906497211775387,
 0.1110102451043963,
 0.11126961483594863,
 0.11282583322526266,
 0.11282583322526266,
 0.11243677862793414,
 0.0963558552716898,
 0.09531837634548049,
 0.09207625470107636,
 0.08701854493580605,
 0.08688886007002986,
 0.08598106600959668,
 0.08779665413046298,
 0.08883413305667229,
 0.08675917520425366,
 0.0852029568149397,
 0.08247957463364028,
 0.08585138114382049,
 0.08624043574114901,
 0.08507327194916353,
 0.08390610815717806,
 0.08429516275450658,
 0.08559201141226816,
 0.08351705355984956,
 0.0815717805732071,
 0.07923745298923618,
 0.07690312540526523,
 0.07392037349241345,
 0.0697704577875762,
 0.06860329399559073,
 0.0700298275191285,
 0.0697704577875762,
 0.0701595123849047,
 0.07210478537154713,
 0.06990014265335237,
 0.06782518480093375,
 0.06704707560627673,
 0.0697704577875762,
 0.0683439242640384,
 0.07132667617689017,
 0.07080793671378552,
 0.07119699131111398,
 0.07262352483465179,
 0.0715860459084425,
 0.0701595123849047,
 0.06925171832447155,
 0.07067825184800935,
 0.07210478537154713,
 0.07586564647905589,
 0.07781091946569837,
 0.07910776812345999,
 0.07871871352613152,
 0.07586564647905589,
 0.07443911295551808,
 0.07093762157956165,
 0.0654908572169628,
 0.06263779016988716,
 0.059914407988587715,
 0.05978472312281158,
 0.06121125664634938,
 0.060692517183244726,
 0.0654908572169628,
 0.06575022694851512,
 0.06432369342497732,
 0.0635455842303203,
 0.063286214498768,
 0.06237842043833486,
 0.05758008040461676,
 0.05887692906237843,
 0.057061340941512134,
 0.057839450136169146,
 0.05965503825703539,
 0.05965503825703539,
 0.05978472312281158,
 0.05978472312281158,
 0.06263779016988716,
 0.06445337829075348,
 0.06730644533782906,
 0.06626896641161978,
 0.06730644533782906,
 0.07093762157956165,
 0.07327194916353261,
 0.0766437556737129,
 0.08247957463364028,
 0.08105304111010248,
 0.08922318765400081,
 0.09259499416418107,
 0.08196083517053562,
 0.11191803916482948,
 0.1252755803397744,
 0.13396446634677736,
 0.13552068473609133,
 0.14485799507197514,
 0.15328751134742577,
 0.14913759564258852,
 0.13305667228634419,
 0.14680326805861763,
 0.1398002853067047,
 0.1400596550382571,
 0.1547140448709636,
 0.1500453897030217,
 0.14563610426663212,
 0.14784074698482688,
 0.1500453897030217,
 0.1451173648035274,
 0.13837375178316694,
 0.13487226040721048,
 0.14369083127998963,
 0.1398002853067047,
 0.12903644144728313,
 0.12644274413175985,
 0.12281156789002726,
 0.12008818570872778,
 0.12358967708468424,
 0.13214887822591107,
 0.14070807936713783,
 0.13837375178316694,
 0.13668784852807678,
 0.1388924912462716,
 0.135261315004539,
 0.13461289067565815,
 0.13396446634677736,
 0.1334457268836727,
 0.1305926598365971,
 0.12851770198417847,
 0.1312410841654779,
 0.1287770717157308,
 0.12942549604461165,
 0.12812864738684995,
 0.12657242899753599,
 0.12760990792374535,
 0.12618337440020752,
 0.12670211386331218,
 0.12631305926598366,
 0.12929581117883546,
 0.12008818570872778,
 0.12073661003760863,
 0.11516016080923355,
 0.11840228245363768,
 0.11334457268836726,
 0.1104915056412917,
 0.12462715601089355,
 0.12786927765529763,
 0.12242251329269874,
 0.12670211386331218,
 0.13072234470237323,
 0.12929581117883546,
 0.12618337440020752,
 0.12890675658150694,
 0.12605368953443133,
 0.13616910906497218,
 0.12838801711840228,
 0.12942549604461165,
 0.13188950849435874,
 0.13603942419919599,
 0.13902217611204773,
 0.14382051614576583,
 0.14382051614576583,
 0.14472831020619895,
 0.14939696537414085,
 0.15147192322655947,
 0.1547140448709636,
 0.1528984567500973,
 0.1553624691998444,
 0.15782648164959148,
 0.15678900272338223,
 0.1571780573207107,
 0.15769679678381535,
 0.15678900272338223,
 0.15678900272338223,
 0.15549215406562059,
 0.15665931785760603,
 0.15925301517312931,
 0.16029049409933863,
 0.16599662819348984,
 0.17351835040850735,
 0.17701984178446376,
 0.17701984178446376,
 0.17650110232135915,
 0.1775385812475684,
 0.1812994423550772,
 0.18648683698612376,
 0.18259629101283884,
 0.18324471534171963,
 0.1793541693684347,
 0.17650110232135915,
 0.16405135520684735,
 0.15886396057580085,
 0.15562183893139672,
 0.15795616651536767,
 0.1618467124886526,
 0.16132797302554794,
 0.16314356114641423,
 0.16534820386460905,
 0.16534820386460905,
 0.15977175463623397,
 0.15510309946829212,
 0.15743742705226307,
 0.15795616651536767,
 0.15367656594475423,
 0.15393593567630656,
 0.1560108935287252,
 0.15588120866294905,
 0.15380625081053043,
 0.15406562054208275,
 0.15406562054208275,
 0.15030475943457403,
 0.14835948644793154,
 0.14835948644793154,
 0.14745169238749836,
 0.14758137725327455,
 0.14446894047464667,
 0.1427830372195565,
 0.1462845285955129,
 0.14693295292439376,
 0.1470626377901699,
 0.14485799507197514,
 0.13850343664894307,
 0.13539099987031514,
 0.13318635715212038,
 0.13487226040721048,
 0.13681753339385297,
 0.14135650369601868,
 0.1445986253404228,
 0.14096744909869016,
 0.1398002853067047,
 0.1417455582933472,
 0.1417455582933472,
 0.14109713396446635,
 0.14096744909869016,
 0.13941123070937625,
 0.1342238360783297,
 0.13461289067565815,
 0.13759564258850995,
 0.13837375178316694,
 0.141615873427571,
 0.14382051614576583,
 0.14719232265594603,
 0.14952665023991699,
 0.14719232265594603,
 0.14382051614576583,
 0.14304240695110879,
 0.14550641940085593,
 0.14096744909869016,
 0.13837375178316694,
 0.13915186097782392,
 0.1399299701724809,
 0.14044870963558556,
 0.13733627285695763,
 0.13967060044092858,
 0.13474257554143435,
 0.13098171443392556,
 0.13150045389703027,
 0.12968486577616398,
 0.1277395927895215,
 0.12825833225262614,
 0.13033329010504477,
 0.1298145506419401,
 0.12760990792374535,
 0.12203345869537027,
 0.11334457268836726,
 0.11386331215147191,
 0.1141226818830243,
 0.11062119050706784,
 0.10958371158085853,
 0.11347425755414345,
 0.11477110621190509,
 0.10997276617818705,
 0.10789780832576842,
 0.10841654778887308,
 0.10673064453378292,
 0.1104915056412917,
 0.10984308131241086,
 0.1075087537284399,
 0.10660095966800673,
 0.10595253533912594,
 0.10361820775515498,
 0.1018026196342869,
 0.10063545584230318,
 0.10128388017118403,
 0.10608222020490213,
 0.10945402671508234,
 0.11554921540656207,
 0.11567890027233821,
 0.10374789262093118,
 0.10517442614446895,
 0.102840098560498,
 0.10880560238620154,
 0.10556348074179747,
 0.1104915056412917,
 0.11204772403060567,
 0.10958371158085853,
 0.10660095966800673,
 0.10997276617818705,
 0.11178835429905329,
 0.10815717805732075,
 0.1111399299701725,
 0.11178835429905329,
 0.11295551809103879,
 0.1104915056412917,
 0.1174944883932045,
 0.12449747114511736,
 0.1305926598365971,
 0.1287770717157308,
 ...]
```

In [346…

```
X = np.asarray(X)
y = np.asarray(y)
```

In [347…

```
X.shape , y.shape
```

Out[347…

```
((2915, 1), (2915,))
```

In [348…

```
split = int(0.7 * len(X))
X_train = X[:split]
y_train = y[:split]
X_test = X[split:]
y_test = y[split:]
```

In [349…

```
X_train.shape , y_train.shape, X_test.shape, y_test.shape
```

Out[349…

```
((2040, 1), (2040,), (875, 1), (875,))
```

In [350…

```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
X_train.shape, X_test.shape
```

Out[350…

```
((2040, 1, 1), (875, 1, 1))
```

In [351…

```
X_train
```

Out[351…

```
array([[[0.12877707]],

       [[0.12877707]],

       [[0.12929581]],

       ...,

       [[0.10361821]],

       [[0.1054338 ]],

       [[0.10815718]]])
```

In [352…

```
X_train.shape
```

Out[352…

```
(2040, 1, 1)
```

In [353…

```
inputs = keras.layers.Input(shape=(X_train.shape[1], X_train.shape[2]))
x = keras.layers.LSTM(150, return_sequences= True)(inputs)
x = keras.layers.Dropout(0.3)(x)
x = keras.layers.LSTM(150, return_sequences=True)(x)
x = keras.layers.Dropout(0.3)(x)
x = keras.layers.LSTM(150)(x)
outputs = keras.layers.Dense(1, activation='linear')(x)

model = keras.Model(inputs=inputs, outputs=outputs)
model.compile(optimizer='adam', loss="mse")
model.summary()
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_3 (InputLayer)         [(None, 1, 1)]            0         
_________________________________________________________________
lstm_6 (LSTM)                (None, 1, 150)            91200     
_________________________________________________________________
dropout_4 (Dropout)          (None, 1, 150)            0         
_________________________________________________________________
lstm_7 (LSTM)                (None, 1, 150)            180600    
```

```
_________________________________________________________________
dropout_5 (Dropout)          (None, 1, 150)            0
_________________________________________________________________
lstm_8 (LSTM)                (None, 150)               180600
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 151
=================================================================
Total params: 452,551
Trainable params: 452,551
Non-trainable params: 0
_________________________________________________________________
```

In [354…

```
X_train.shape, y_train.shape
```

Out[354…

```
((2040, 1, 1), (2040,))
```

In [355…

```
history = model.fit(
    X_train, y_train,
    epochs = 20,
    batch_size = 32,
    validation_split = 0.2
)
```

```
Epoch 1/20
51/51 [==============================] - 3s 51ms/step - loss: 0.0030 - val_loss: 0.0057
Epoch 2/20
51/51 [==============================] - 1s 11ms/step - loss: 0.0010 - val_loss: 4.1449e
-05
Epoch 3/20
51/51 [==============================] - 1s 11ms/step - loss: 6.6296e-05 - val_loss: 2.5
949e-05
Epoch 4/20
51/51 [==============================] - 1s 11ms/step - loss: 4.0457e-05 - val_loss: 2.4
025e-05
Epoch 5/20
51/51 [==============================] - 0s 10ms/step - loss: 3.9753e-05 - val_loss: 3.7
790e-05
Epoch 6/20
51/51 [==============================] - 1s 10ms/step - loss: 4.1150e-05 - val_loss: 2.5
174e-05
Epoch 7/20
51/51 [==============================] - 0s 8ms/step - loss: 3.2069e-05 - val_loss: 4.03
90e-05
Epoch 8/20
51/51 [==============================] - 0s 9ms/step - loss: 3.2752e-05 - val_loss: 2.46
66e-05
Epoch 9/20
51/51 [==============================] - 0s 8ms/step - loss: 3.4260e-05 - val_loss: 2.85
44e-05
Epoch 10/20
51/51 [==============================] - 0s 7ms/step - loss: 3.5984e-05 - val_loss: 2.34
59e-05
Epoch 11/20
51/51 [==============================] - 0s 7ms/step - loss: 3.1575e-05 - val_loss: 2.97
29e-05
Epoch 12/20
51/51 [==============================] - 0s 8ms/step - loss: 3.0410e-05 - val_loss: 3.67
40e-05
Epoch 13/20
51/51 [==============================] - 0s 9ms/step - loss: 3.1194e-05 - val_loss: 2.87
55e-05
Epoch 14/20
51/51 [==============================] - 0s 8ms/step - loss: 2.9855e-05 - val_loss: 2.99
55e-05
```

```
Epoch 15/20
51/51 [==============================] - 0s 7ms/step - loss: 3.0825e-05 - val_loss: 2.44
23e-05
Epoch 16/20
51/51 [==============================] - 0s 9ms/step - loss: 2.8862e-05 - val_loss: 2.40
80e-05
Epoch 17/20
51/51 [==============================] - 0s 8ms/step - loss: 3.6636e-05 - val_loss: 2.47
57e-05
Epoch 18/20
51/51 [==============================] - 0s 8ms/step - loss: 3.3046e-05 - val_loss: 2.37
31e-05
Epoch 19/20
51/51 [==============================] - 0s 8ms/step - loss: 2.9455e-05 - val_loss: 2.38
31e-05
Epoch 20/20
51/51 [==============================] - 0s 8ms/step - loss: 3.6684e-05 - val_loss: 4.77
91e-05
```

In [356...
```
predicted_LSTM = model.predict(X)
```

In [357...
```
predicted_LSTM
```

Out[357...
```
array([[0.12406403],
       [0.12406403],
       [0.12458012],
       ...,
       [0.9513022 ],
       [0.9426287 ],
       [0.9408232 ]], dtype=float32)
```

In [358...
```
predicted.shape
```

Out[358...
```
(2915, 1)
```

In [359...
```
test_predicted_LSTM = []

for i in predicted_LSTM:
  test_predicted_LSTM.append(i[0])
```

In [360...
```
len(test_predicted_LSTM)
```

Out[360...
```
2915
```

In [361...
```
df_predicao_LSTM = pd.DataFrame(columns = ['data' , 'Fechamento', 'Fechamento_Predito']
```

In [362...
```
df_predicao_LSTM
```

Out[362...

data	Fechamento	Fechamento_Predito

In [363...
```
df_predicao_LSTM['data'] = mc_df[1:]['data']
```

In [364...
```
df_predicao_LSTM
```

Out[364...

	data	Fechamento	Fechamento_Predito
data			

	data	Fechamento	Fechamento_Predito
data			
2008-09-22	2008-09-22	NaN	NaN
2008-09-26	2008-09-26	NaN	NaN
2008-09-30	2008-09-30	NaN	NaN
2008-10-02	2008-10-02	NaN	NaN
2008-10-03	2008-10-03	NaN	NaN
...	...	...	...
2021-02-24	2021-02-24	NaN	NaN
2021-02-25	2021-02-25	NaN	NaN
2021-02-26	2021-02-26	NaN	NaN
2021-03-01	2021-03-01	NaN	NaN
2021-03-02	2021-03-02	NaN	NaN

2915 rows × 3 columns

In [365…

```
Fechamento_Scaled = []
for i in training_set_scaled:
  Fechamento_Scaled.append(i[0])
```

In [366…

```
len(Fechamento_Scaled)
```

Out[366…

```
2916
```

In [367…

```
df_predicao_LSTM
```

Out[367…

	data	Fechamento	Fechamento_Predito
data			
2008-09-22	2008-09-22	NaN	NaN
2008-09-26	2008-09-26	NaN	NaN
2008-09-30	2008-09-30	NaN	NaN
2008-10-02	2008-10-02	NaN	NaN
2008-10-03	2008-10-03	NaN	NaN
...	...	...	...
2021-02-24	2021-02-24	NaN	NaN
2021-02-25	2021-02-25	NaN	NaN
2021-02-26	2021-02-26	NaN	NaN
2021-03-01	2021-03-01	NaN	NaN
2021-03-02	2021-03-02	NaN	NaN

2915 rows × 3 columns

In [368…
```
df_predicao_LSTM['Fechamento'] = Fechamento_Scaled[1:]
```

In [369…
```
df_predicao_LSTM
```

Out[369…

	data	Fechamento	Fechamento_Predito
data			
2008-09-22	2008-09-22	0.128777	NaN
2008-09-26	2008-09-26	0.129296	NaN
2008-09-30	2008-09-30	0.127610	NaN
2008-10-02	2008-10-02	0.128777	NaN
2008-10-03	2008-10-03	0.115938	NaN
...	...	...	...
2021-02-24	2021-02-24	0.995461	NaN
2021-02-25	2021-02-25	0.997536	NaN
2021-02-26	2021-02-26	0.987550	NaN
2021-03-01	2021-03-01	0.985475	NaN
2021-03-02	2021-03-02	0.988588	NaN

2915 rows × 3 columns

In [370…
```
df_predicao_LSTM['Fechamento_Predito'] = test_predicted_LSTM
```

In [371…
```
df_predicao_LSTM
```

Out[371…

	data	Fechamento	Fechamento_Predito
data			
2008-09-22	2008-09-22	0.128777	0.124064
2008-09-26	2008-09-26	0.129296	0.124064
2008-09-30	2008-09-30	0.127610	0.124580
2008-10-02	2008-10-02	0.128777	0.122903
2008-10-03	2008-10-03	0.115938	0.124064
...	...	...	...
2021-02-24	2021-02-24	0.995461	0.944320
2021-02-25	2021-02-25	0.997536	0.949502
2021-02-26	2021-02-26	0.987550	0.951302
2021-03-01	2021-03-01	0.985475	0.942629
2021-03-02	2021-03-02	0.988588	0.940823

2915 rows × 3 columns

In [372…

```
interactive_plot(df_predicao_LSTM, "LSTM - CCMFUT Fechamento e CCMFUT Fechamento Predit
```

In [373…

```
#Cálculo do erro
mse = mean_squared_error(df_predicao_LSTM['Fechamento'], df_predicao_LSTM['Fechamento_P
print('MSE: '+str(mse))
mae = mean_absolute_error(df_predicao_LSTM['Fechamento'], df_predicao_LSTM['Fechamento_
print('MAE: '+str(mae))
rmse = math.sqrt(mean_squared_error(df_predicao_LSTM['Fechamento'], df_predicao_LSTM['F
print('RMSE: '+str(rmse))
```

```
MSE: 0.00011615309602060521
MAE: 0.0069011389543802695
RMSE: 0.010777434575102055
```

In [ ]: