Modelo Futuros Mini Ibovespa - Dados Históricos

O Mercado Futuro é o ambiente onde você pode ganhar com a alta ou baixa de um determinado ativo, seja ele uma commodity (Milho, Café, Boi Gordo), uma moeda (como o dólar), um Índice (Bovespa, Índice S&P 500) ou mesmo uma taxa de juros. Nele, são negociados contratos futuros.

O mini índice é um contrato futuro derivado do Índice Bovespa, ou seja, é um ativo que tem como base o sobe e desce desse índice. Como esse tipo de operação envolve **risco considerável** e **oscilações frequentes no mercado**, ela é indicada apenas para aqueles que se encaixam no perfil de investidor arrojado.

Neste trabalho iremos implementar uma RNNs para realizar a predição diária do Mini Índice da Ibovespa.

O dataset "**FuturosMiniBovespa.csv**" possui informações dispostas em colunas :

- **Date**: Data das operações na bolsa (diária)
- **Close**: Valor de Fechamento do Índice da Ibovespa (no dia)
- **Open**: Valor da Abertura do Índice da Ibovespa (no dia)
- **High**: Valor máximo do Índice da Ibovespa (no dia)
- **Low**: Valor mínimo do Índice da Ibovespa (no dia)
- **Vol**: Volume de contratos negociados (no dia)

Bibliotecas

```
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM
import plotly.graph_objects as go
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Carregando os dados

Vamos começar lendo o arquivo FuturosMiniBovespa.csv em um dataframe do pandas, mas antes vamos dar uma olhadinha no gráfico de variação do último mês do índice Ibovespa.

✓ 0s completed at 11:26 AM

```
                    ])

fig.update_layout(xaxis_rangeslider_visible=False)
fig.show()
```

Rede Neural Recorrente (RNN)

Antes de avançar para LSTM, primeiro vamos introduzir o conceito de Redes Recorrentes. Elas são redes utilizadas para reconhecer padrões quando os resultados do passado influenciam no resultado atual. Um exemplo disso são as séries temporais, em que a ordem dos dados é muito

Rede LSTM

Uma rede LSTM tem origem em uma RNN (Rede Neural Recorrente). Mas ela resolve o problema de memória mudando sua arquitetura.

Nesta nova arquitetura, cada neurônio possui 3 gates, cada um com uma função diferente. São eles:

- Input Gate
- Output Gate
- Forget Gate

Agora, um neurônio LSTM recebe entradas de seu estado anterior, assim como ocorria na Rede Recorrente:

Agora vamos ler o arquivo do período desejável

```
DataSet=pd.read_csv('FuturosEthereum-treino.csv')
DataSet=DataSet.dropna()
DataSet.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2015-08-07	2.831620	3.536610	2.521120	2.772120	2.772120	164329.0
1	2015-08-08	2.793760	2.798810	0.714725	0.753325	0.753325	674188.0
2	2015-08-09	0.706136	0.879810	0.629191	0.701897	0.701897	532170.0
3	2015-08-10	0.713989	0.729854	0.636546	0.708448	0.708448	405283.0

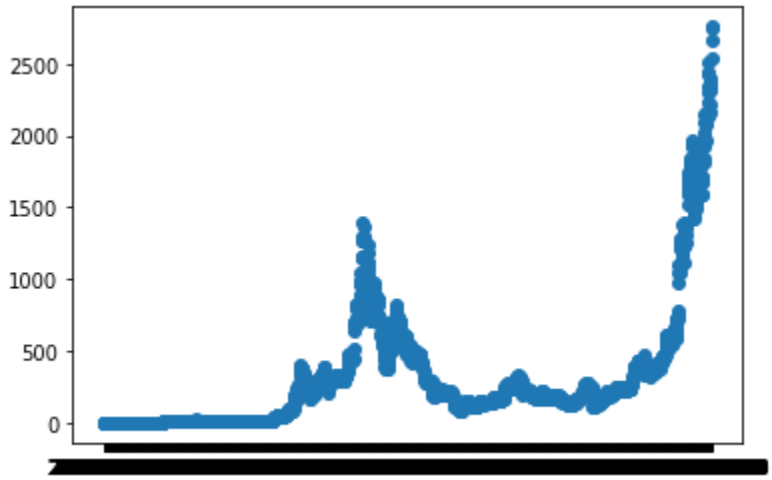
75%	359.261754	371.923491	348.044752	359.041008	359.041008	6.5709
max	2757.734131	2797.972412	2728.169922	2773.207031	2773.207031	6.0733

Inicialmente iremos criar uma RNN baseada apenas no Valor de Abertura

```
plt.scatter(DataSet['Date'],DataSet['Open'],)
plt.show()

base_treinamento = DataSet.iloc[:, 1:2].values

#DataSet.drop(['Date','Close','High','Low', 'Volume'],axis=1,inplace=True)
```




```
base_treinamento

     array([[2.83162000e+00],
```

Definição dos previsores

```
previsores = []
preco_real = []
NRecursao = 90
DataSetLen = len(DataScaled)
print(DataSetLen)
```

```
2090
```

```
for i in range(NRecursao, DataSetLen):
    previsores.append(DataScaled[i-NRecursao:i,0])
    preco_real.append(DataScaled[i,0])

previsores, preco_real = np.array(previsores), np.array(preco_real)
```

```
previsores.shape
```

```
(2000, 90)
```

```
regressor.add(Dropout(0.3))

# Cada Oculta 3
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.3))

# Camada de Saída
regressor.add(Dense(units = 1, activation = 'linear'))
```

Construindo a Rede

```
regressor.compile(optimizer = 'rmsprop', loss = 'mean_squared_error',
                  metrics = ['mean_absolute_error'])
regressor.fit(previsores, preco_real, epochs = 100, batch_size = 32)
```

```
Epoch 1/100
63/63 [==============================] - 18s 182ms/step - loss: 0.0106 - mean_
```

```
Epoch 22/100
63/63 [==============================] - 11s 175ms/step - loss: 9.3160e-04 - r
Epoch 23/100
63/63 [==============================] - 11s 174ms/step - loss: 9.6013e-04 - r
Epoch 24/100
63/63 [==============================] - 11s 178ms/step - loss: 0.0012 - mean_
Epoch 25/100
63/63 [==============================] - 11s 176ms/step - loss: 9.7575e-04 - r
Epoch 26/100
63/63 [==============================] - 11s 175ms/step - loss: 0.0010 - mean_
Epoch 27/100
63/63 [==============================] - 11s 177ms/step - loss: 0.0011 - mean_
Epoch 28/100
63/63 [==============================] - 11s 177ms/step - loss: 0.0012 - mean_
Epoch 29/100
```

```
print(RNN.mean())
print(previsoes.mean())
print(preco_real_teste.mean())
```

```
2096.7557163208708
2823.6003
```