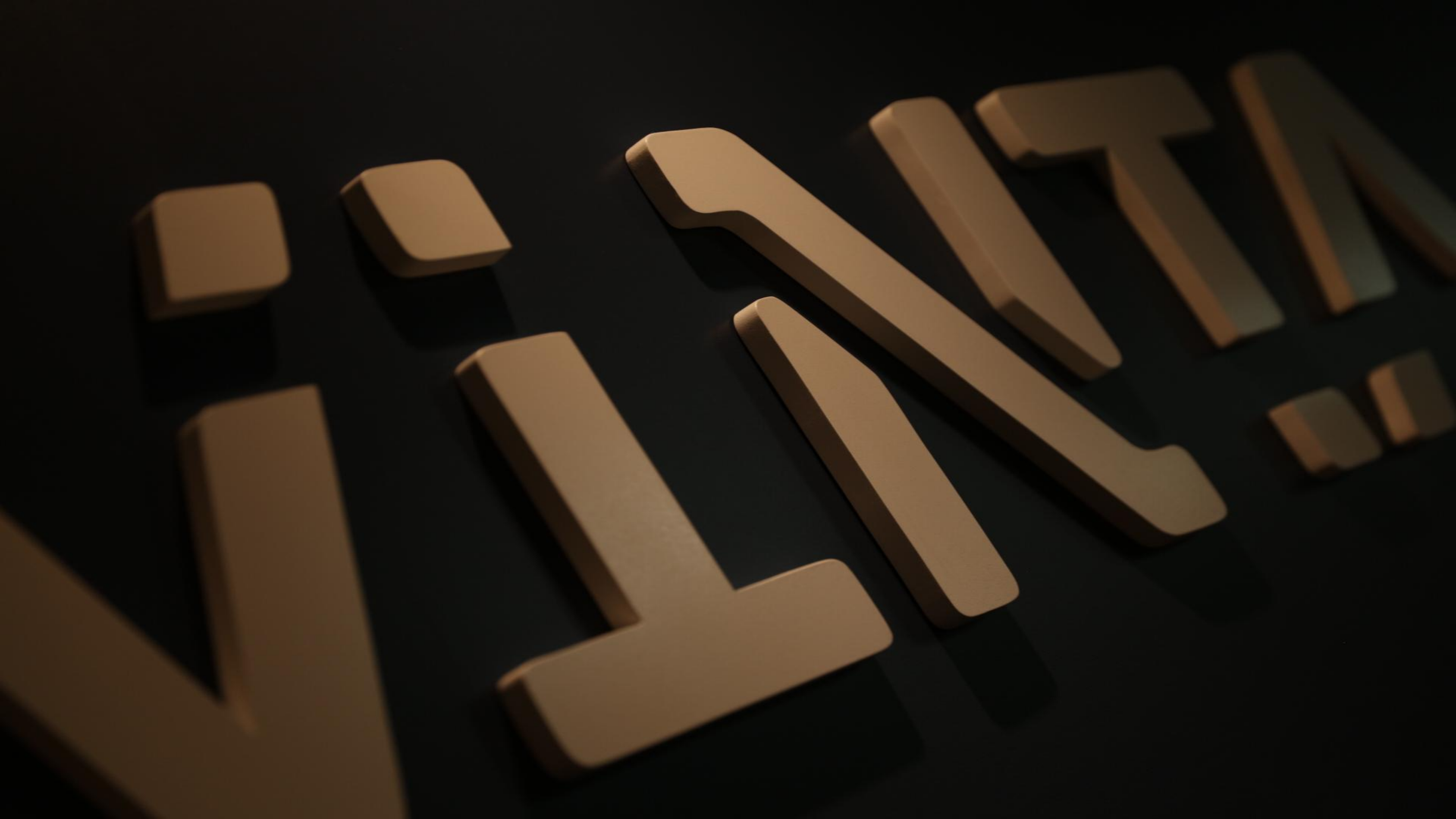
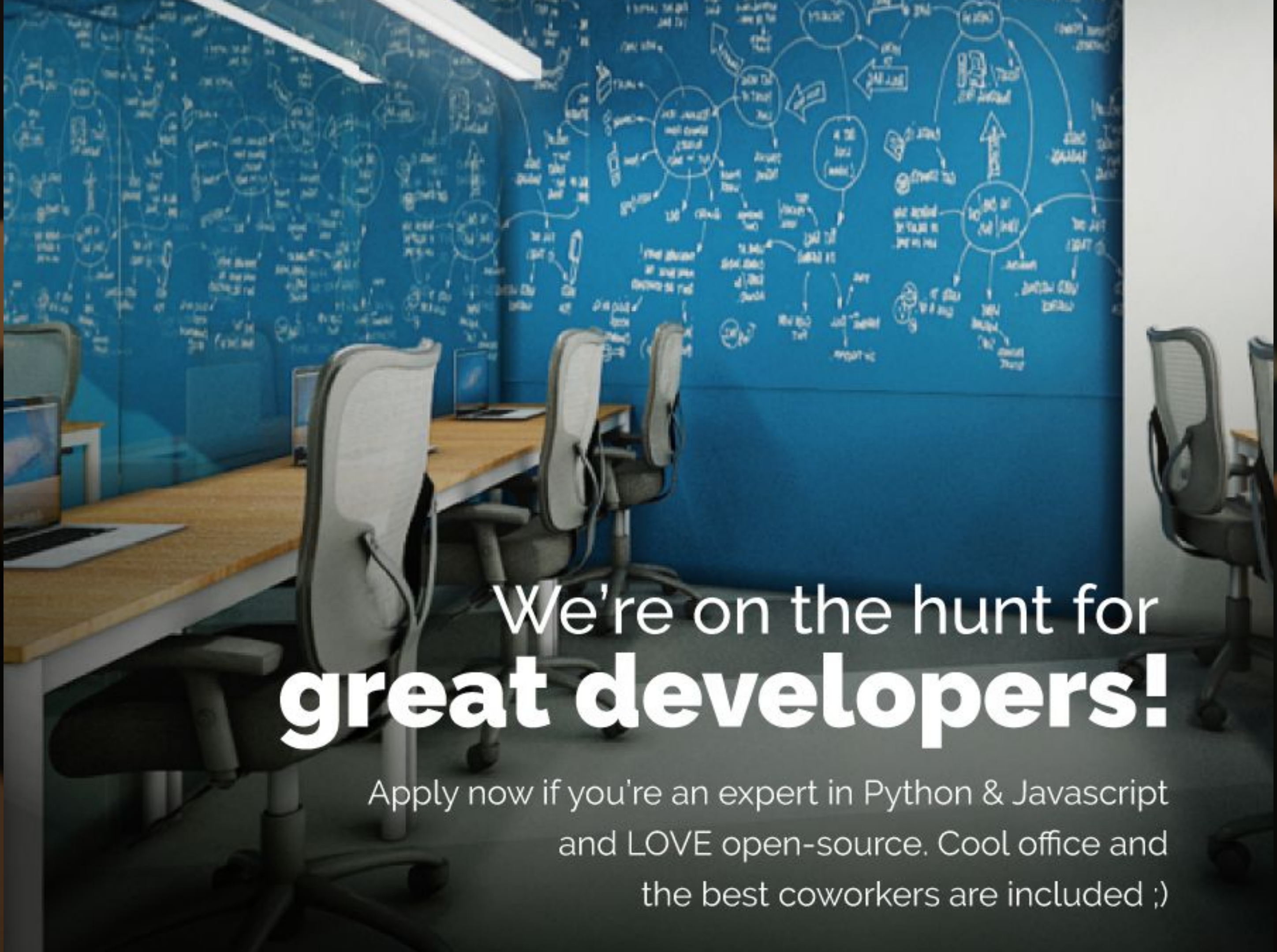
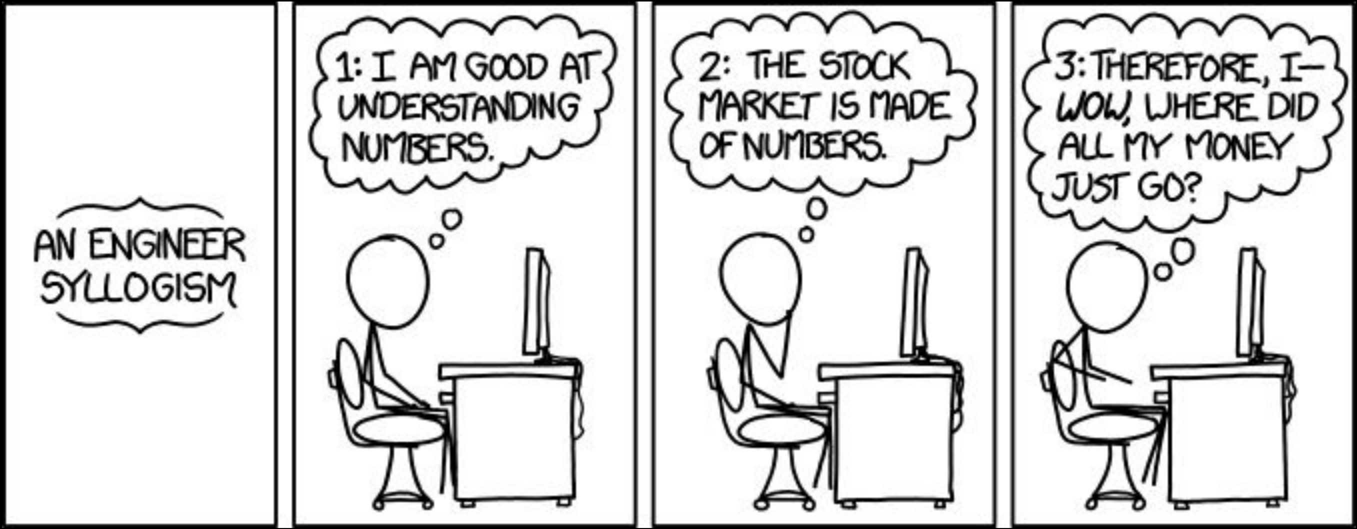
Previsão de séries temporais com pydata e inteligência artificial

Rebeca Sarai

- Recife
- Estudante de Engenharia da Computação - UPE/POLI
- Torcedora do ~~melhor time de Pernambuco~~ Náutico
- Organizadora do Django Girls Recife
- Motociclista




We're on the hunt for
great developers!
Apply now if you're an expert in Python & Javascript and LOVE open-source. Cool office and the best coworkers are included ;)


AN ENGINEER SYLLOGISM
1: I AM GOOD AT UNDERSTANDING NUMBERS.
2: THE STOCK MARKET IS MADE OF NUMBERS.
3: THEREFORE, I— *WOW*, WHERE DID ALL MY MONEY JUST GO?

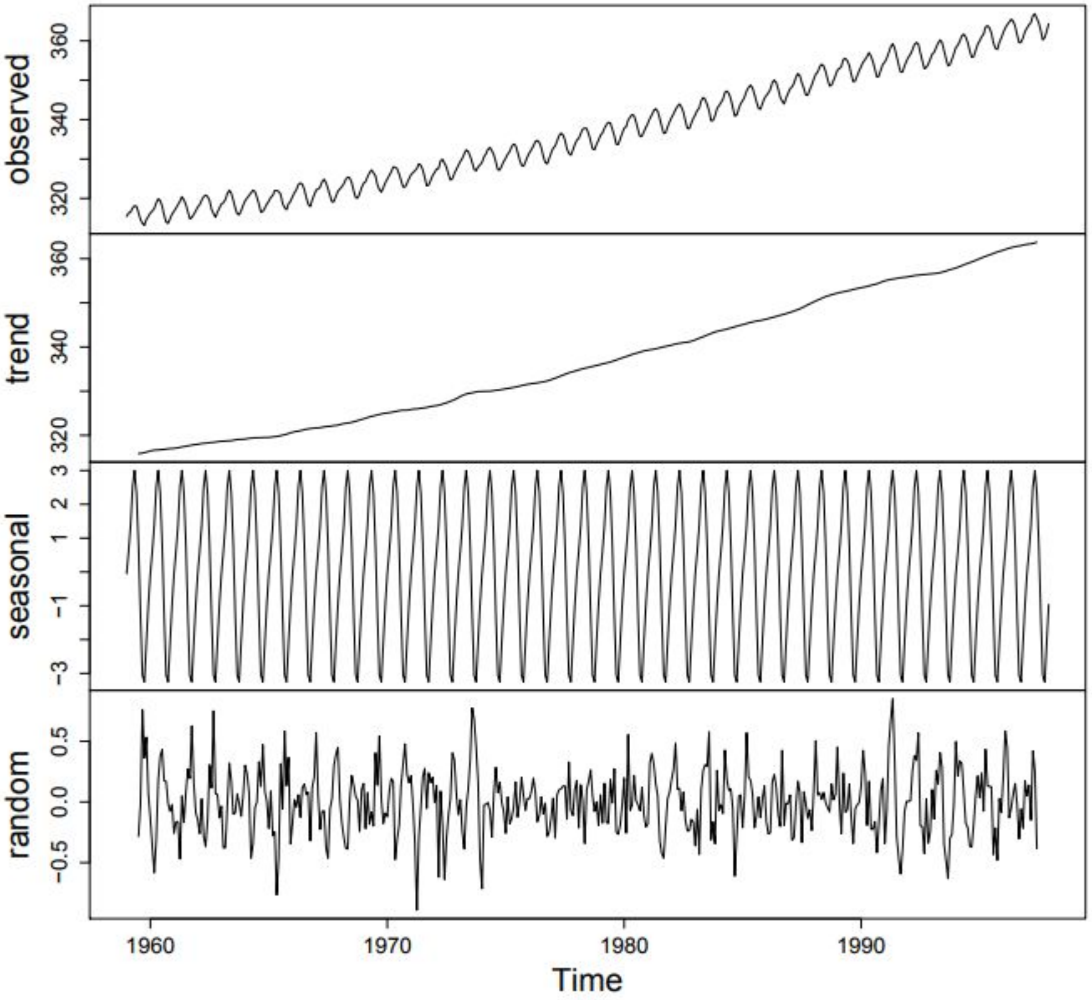
Séries Temporais

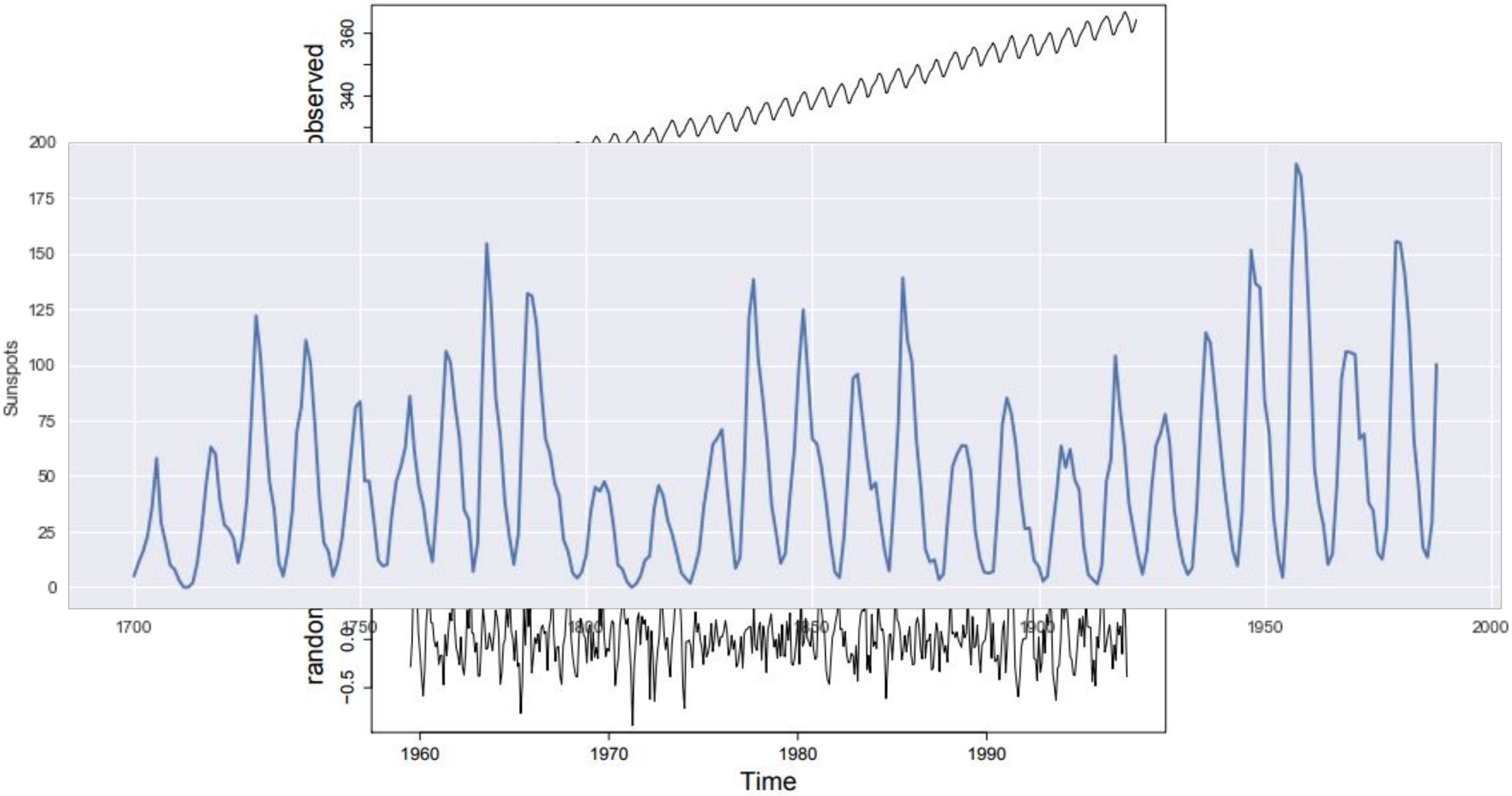
É uma **sequência** de medidas da mesma variável coletadas **ordenadamente**. Na maioria das vezes, as medições são feitas em **intervalos de tempo regulares**.


THE FLASH NEWS
Imaginem que essa linha
é o tempo, certo?
THE FLASH BRASIL
www.THEFLASHBR.com
Eu deixei ela assim.

Apresentam comportamento **dinâmico**

Podem possuir: **Tendência**, **Ciclos**, **Sazonalidade**, **Outlier**

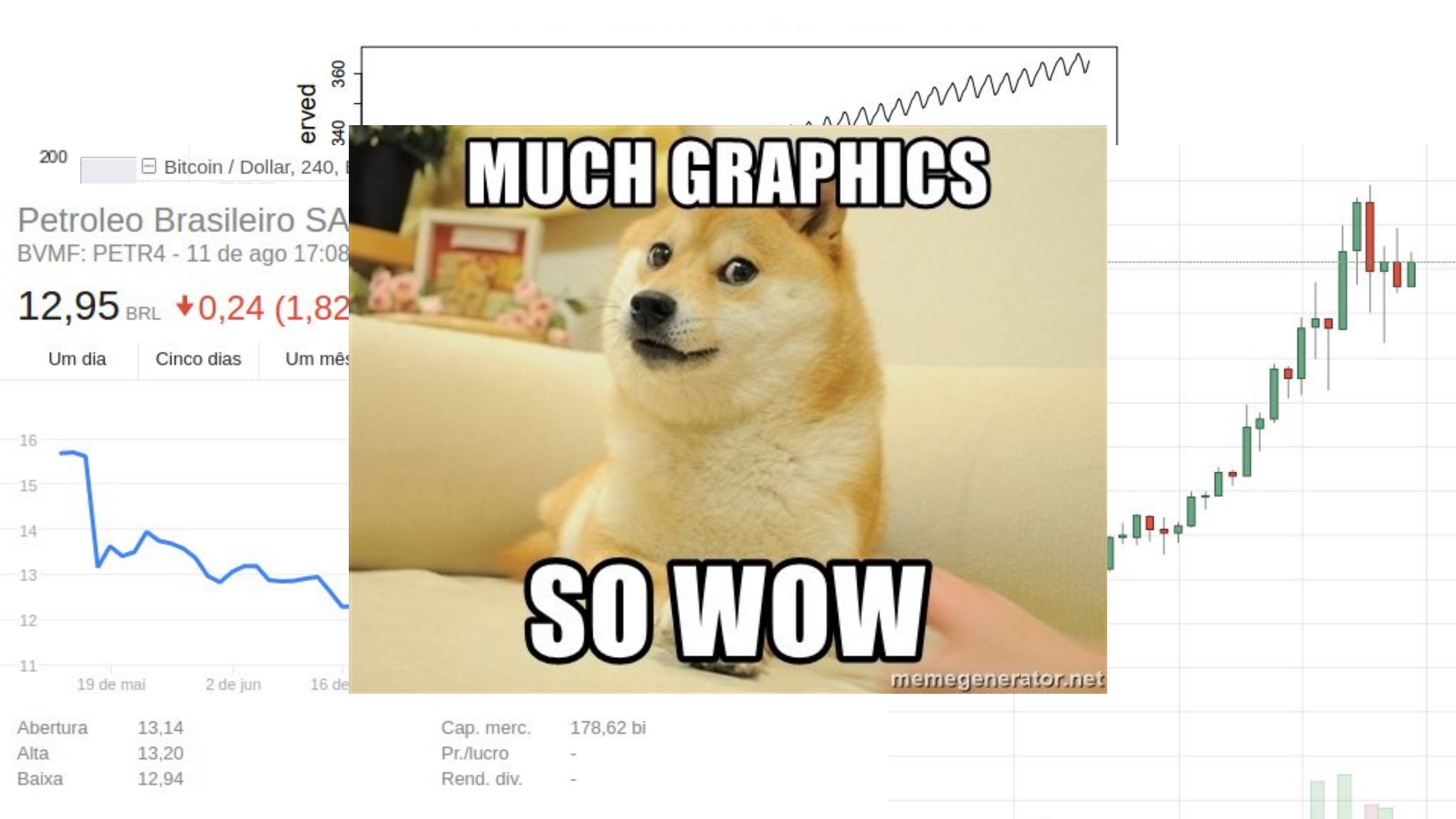

observed
trend
seasonal
random
Time
1960
1970
1980
1990



observed
360
340
Sunspots
200
175
150
125
100
75
50
25
0
1700
1950
2000
random
0.0
−0.5
1960
1970
1980
1990
Time

erved
360
340
Sunspots
200
175
150
125
100
75
50
25
0
1
Bitcoin / Dollar, 240, BITFINEX
O 3962.0 H 4037.7 L 3961.9 C 4015.9
Vol (20)
981 6K


erved
360
340
200
Bitcoin / Dollar, 240, BITFINEX
O 3962.0 H 4037.7 L 3961.9 C 4015.9
Petroleo Brasileiro SA Petrobras Preference Shares
BVMF: PETR4 - 11 de ago 17:08 BRT
12,95 BRL
0,24 (1,82%)
Um dia
Cinco dias
Um mês
Três meses
Um ano
Cinco anos
máx
16
15
14
13
12
11
19 de mai
2 de jun
16 de jun
30 de jun
14 de jul
28 de jul
11 de ago
Abertura 13,14
Alta 13,20
Baixa 12,94
Cap. merc. 178,62 bi
Pr./lucro -
Rend. div. -



360
340
erved
200
Bitcoin / Dollar, 240,
Petroleo Brasileiro SA
BVMF: PETR4 - 11 de ago 17:08
12,95 BRL
0,24 (1,82
Um dia
Cinco dias
Um mês
16
15
14
13
12
11
19 de mai
2 de jun
16 de
MUCH GRAPHICS
SO WOW
memegenerator.net
Abertura 13,14
Alta 13,20
Baixa 12,94
Cap. merc. 178,62 bi
Pr./lucro -
Rend. div. -

A análise de séries temporais tem por meta

Entender seu comportamento estatístico
Encontrar um **modelo** que se adeque ao **padrão gerador**
Explicar como o **passado** afeta o **futuro**

Ferramentas

matplotlib

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

NumPy

jupyter

PROPHET

TensorFlow ™

scikit learn

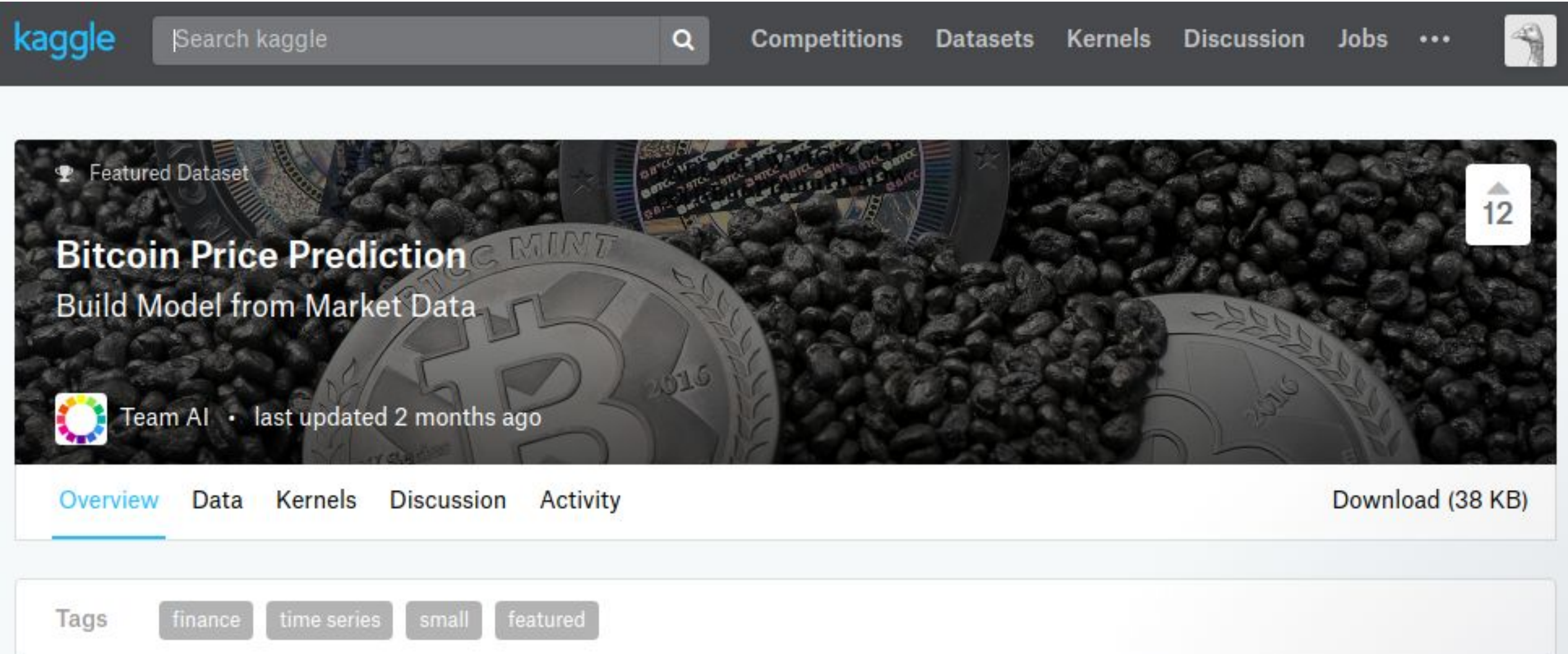
Keras

NeuPy

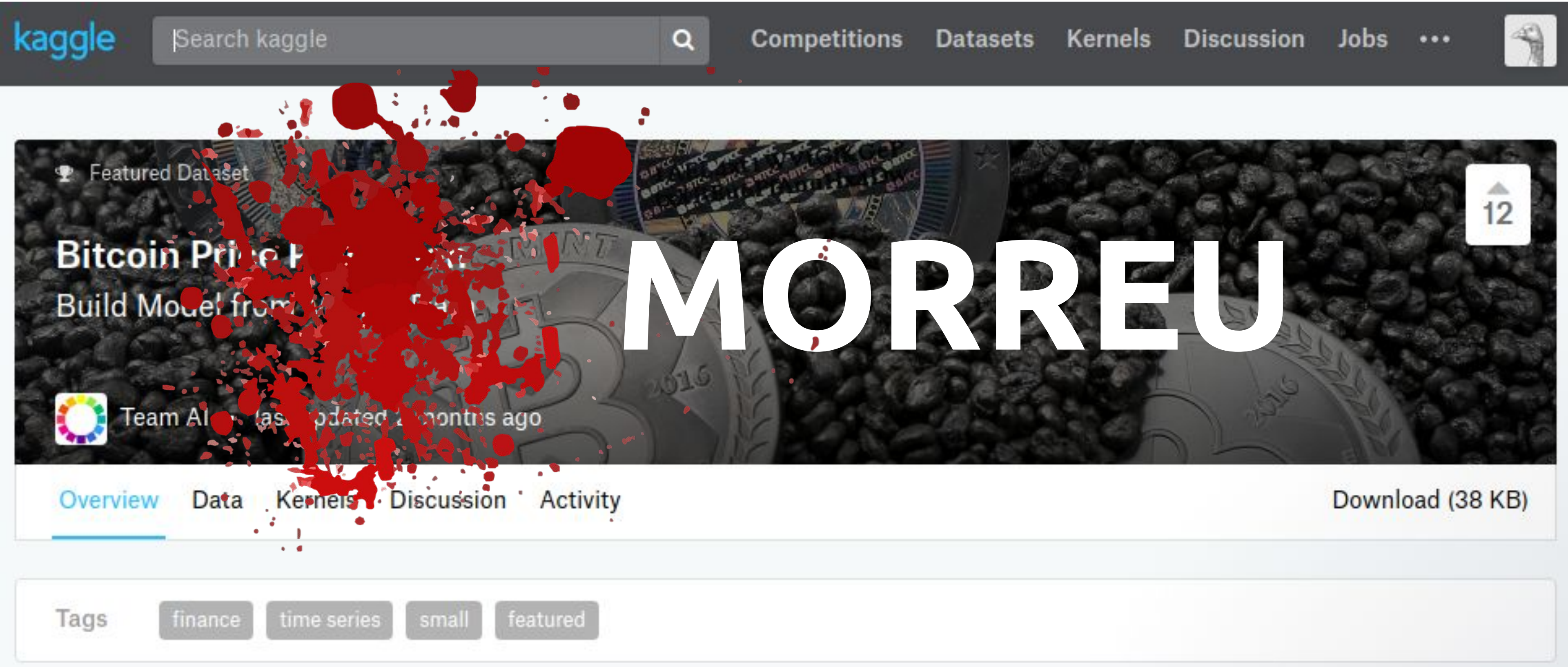
Neural Networks in Python

theano

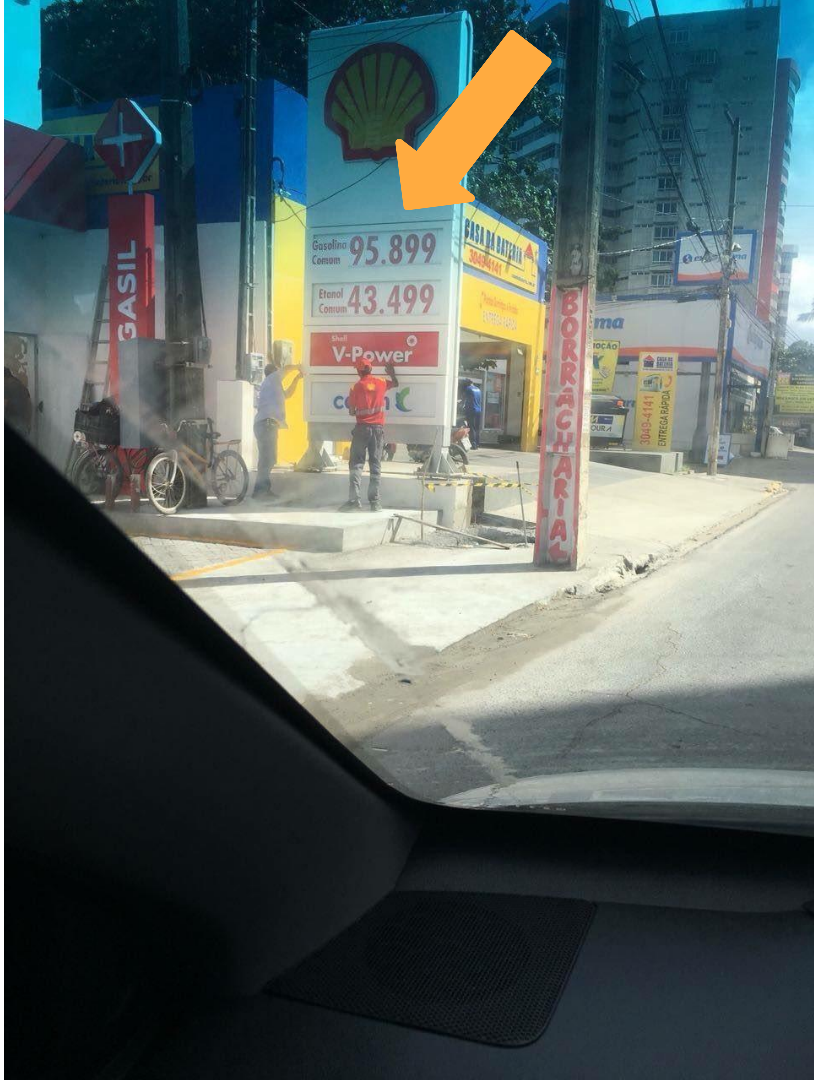
Qual será o preço do bitcoin?

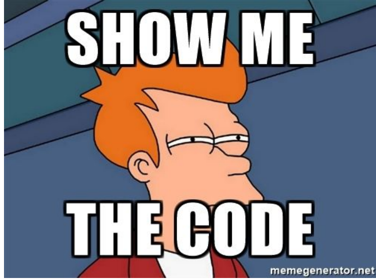


Qual será o preço do bitcoin?



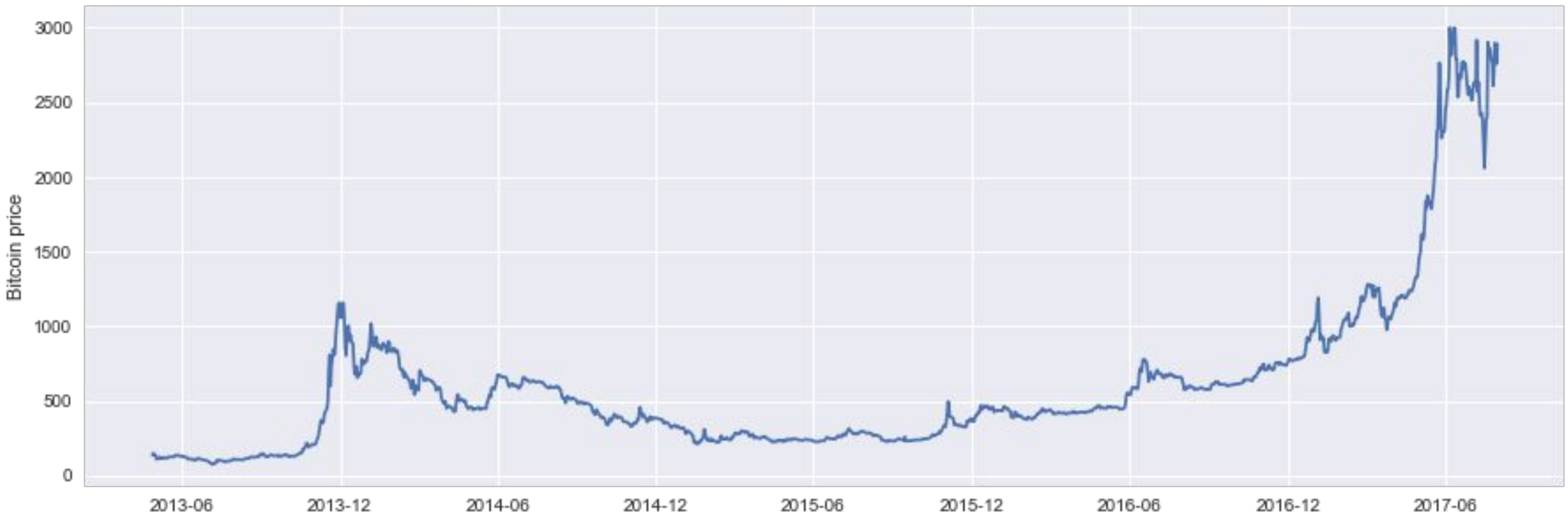
Qual será o preço da gasolina?





In [7]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
from datetime import datetime
from pandas_datareader.data import DataReader
from dateutil.parser import parse
from datetime import datetime

%matplotlib inline

data_location = 'bitcoin-price-prediction/bitcoin_price_Training - bitcoin_price2013Apr-2017Aug.csv'
raw_price_data = pd.read_csv(data_location)
raw_price_data = raw_price_data[::-1]
date = raw_price_data['Date'].values
date_n = convert(date)
raw_price_data['Date'] = date_n
raw_price_data = raw_price_data.set_index('Date')

plt.figure(figsize=(15,5))
plt.plot(raw_price_data.index, raw_price_data['High'])
plt.ylabel('Bitcoin price');
```



Bitcoin price
3000
2500
2000
1500
1000
500
0
2013-06
2013-12
2014-06
2014-12
2015-06
2015-12
2016-06
2016-12
2017-06

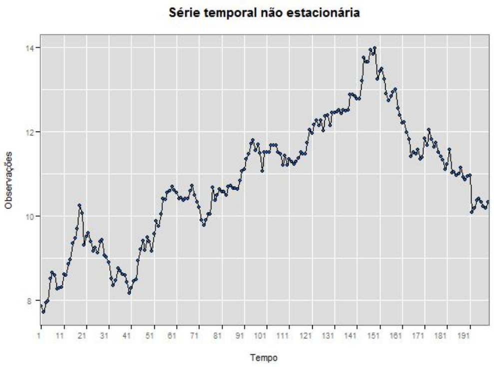
Modelos de Previsão

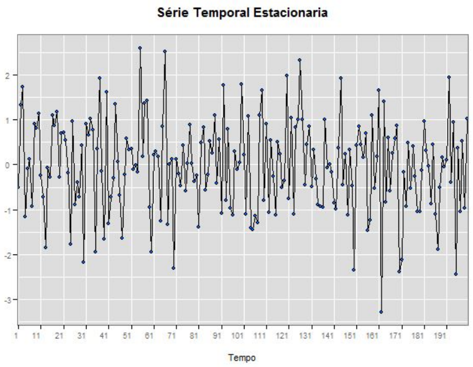
ARIMA

(autoregressive integrated moving average)

ARIMA (autoregressive integrated moving average)

- Captura a autocorrelação através de **AR** e **MA**
- Diferencia a série até ficar **estacionária**
- Lags selecionadas por critérios de informação ou validação cruzada
- Aplica inferência para obter estimativas variáveis latentes





- Modelo Auto-Regressivo (AR)

$$\hat{x}_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots \phi_p x_{t-p} + a_t$$

- Modelo Médias Móveis (MA)

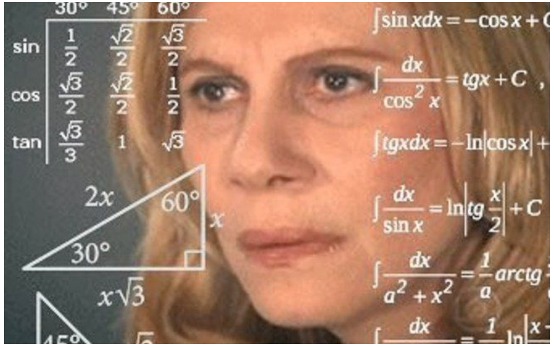
$$x_t = -\theta_1 a_{t-P} - \theta_2 a_{t-P-1} - \ldots - \theta_q a_{t-P-q+1} + a_t$$

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t$$

- Modelo Auto-Regressivo (AR)

$$\hat{x}_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \dots \phi_p x_{t-p} + a_t$$

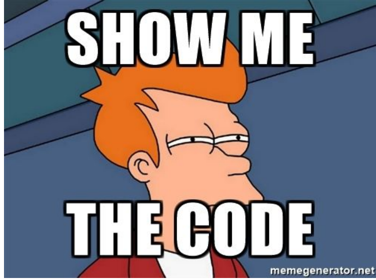
- Modelo Médias Móveis (MA)



$$x_t = -\theta_1 a_{t-P} - \theta_2 a_{t-P-1} - \dots - \theta_q a_{t-P-q+1} + a_t$$

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t$$

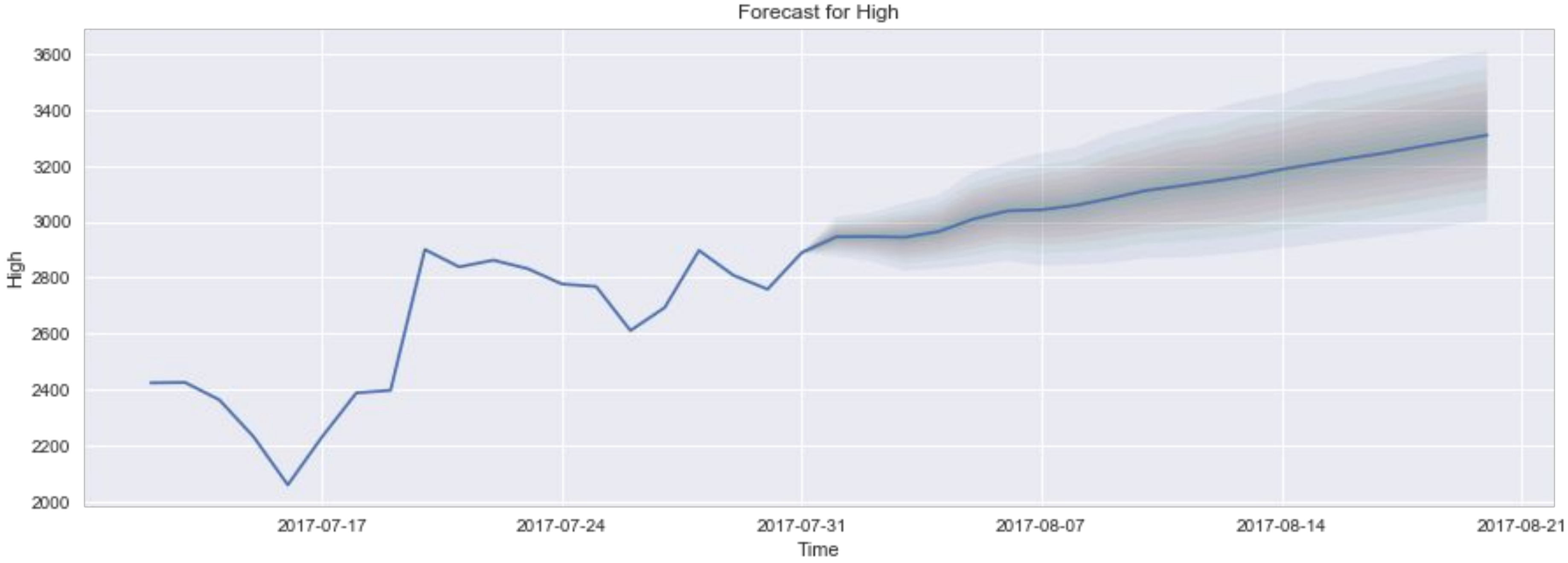
Aplicando o modelo ARIMA




```
raw_price_data = pd.read_csv(data_location)
raw_price_data = raw_price_data[::-1]
date = raw_price_data['Date'].values
date_n = convert(date)
raw_price_data['Date'] = date_n
raw_price_data = raw_price_data.set_index('Date')

model = pf.ARIMA(ar=9, ma=2, data=raw_price_data, target='High')
# The distribution for the time series
model.adjust_prior([0], pf.Exponential())
# maximum likelihood point | is a method of estimating the parameters of a statistical model given observations,
# by finding the parameter values that maximize the likelihood of making the observations given the parameters.
x = model.fit('MLE', nsims=50000)
model.plot_fit(figsize=(15,6))
```


High
Data
ARIMA model
3000
2500
2000
1500
1000
500
0
2013-06
2013-12
2014-06
2014-12
2015-06
2015-12
2016-06
2016-12
2017-06

In [9]:
```
# h = How many steps to forecast ahead
model.plot_predict(h=20,past_values=20,figsize=(15,5))
```



O que realmente aconteceu...

Hoje


Bitcoin / Brazilian Real, 240, FOXBIT
O 28800.00 H 29294.21 L 28400.00 C 28690.00
Vol (20) 19 24
36400.00
36000.00
35600.00
35200.00
34800.00
34400.00
34000.00
33600.00
33200.00
32800.00
32400.00
32000.00
31600.00
31200.00
30800.00
30400.00
30000.00
29600.00
29200.00
28690.00
02:53:02

Limitações

- Os modelos ARIMA dependem de premissas lineares relativas aos dados. Devido à natureza altamente não linear do preço da Bitcoin, não se esperava que apresentasse um bom desempenho.
  - Bitcoin, Dólar, Real, Ações...

Em que problemas o ARIMA funciona?

Forecasting Daily Maximum Surface Ozone Concentrations in Brunei Darussalam—An ARIMA Modeling Approach

Krishan Kumar , A.K. Yadav , M.P. Singh , H. Hassan & V.K. Jain

f ARIMA(1,1,0) Model for Predicting Time
Engine Crawlers

Jeeva JOSE[1], P. Sojan LAL[2]
ent of Computer Applications, BPC College, Piravom,
mputer Sciences, Mahatma Gandhi University, Kottay
vijojeeva@yahoo.co.in, padikkakudy@gmail.com

The Use of an Autoregressive Integrated Moving Average Model for Prediction of the Incidence of Dysentery in Jiangsu, China

Kewei Wang, PhD, Wentao Song, MPH, Jinping Li, MPH, more... Show all authors

First Published April 22, 2016 | Research Article

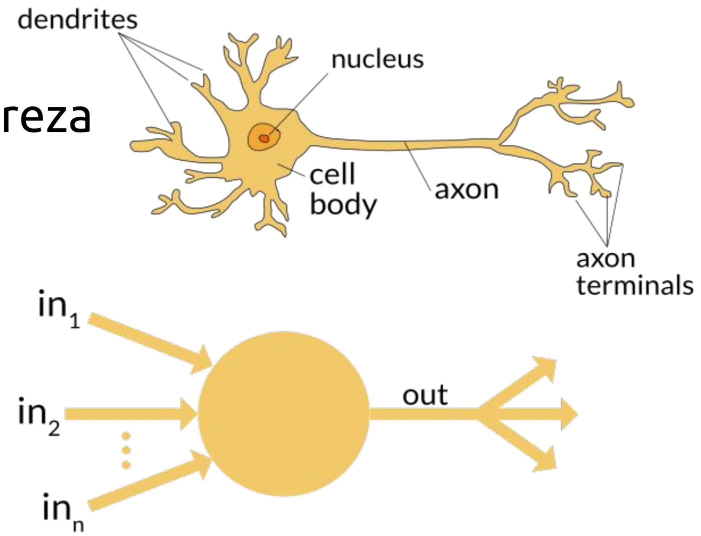
Application of an Autoregressive Integrated Moving Average Model for Predicting the Incidence of Hemorrhagic Fever with Renal Syndrome

Qi Li,* Na-Na Guo, Zhan-Ying Han, Yan-Bo Zhang, Shun-Xiang Qi, Yong-Gang Xu, Ya-Mei Wei, Xu Han, and Ying-Ying Liu

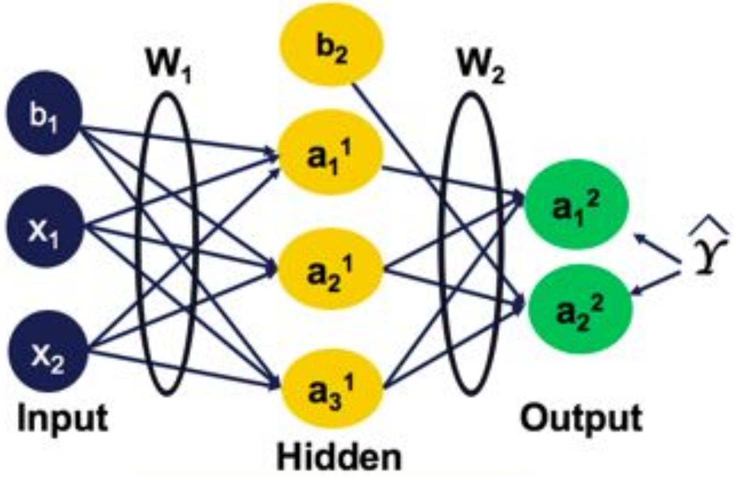
REDES NEURAIS ARTIFICIAIS

Uma estrutura composta por unidades de processamento (neurônios artificiais) interconectadas, tendo cada unidade de processamento uma função de ativação específica

- Inspiração biológica
- Conseguem lidar com problemas de natureza dinâmica e temporal
- Dotadas de laços de realimentação
- Modelos não lineares



- Perceptron
    - Tipo mais simples de rede neural, um classificador linear.
- MLP
    - Rede neural simples com pelo menos uma camada intermediária (oculta)
- RBF
    - Rede que usa funções base radiais como funções de ativação.

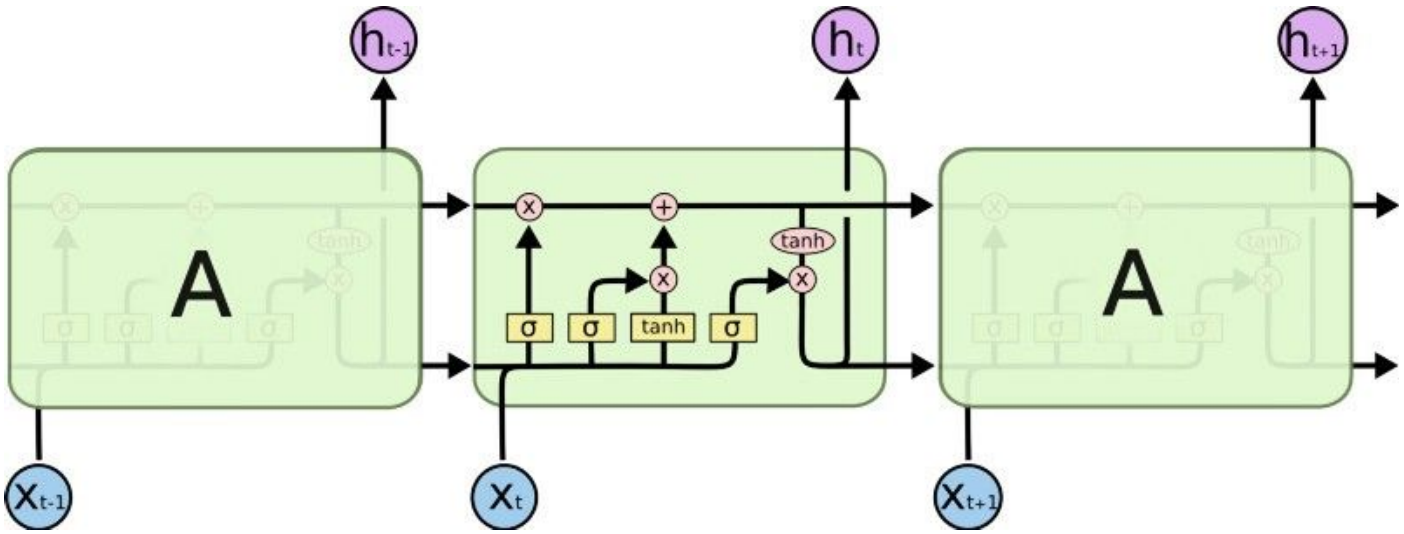


x	y
(0.71, 0.91)	1
(0.25, 0.44)	0
(0.34, 0.61)	1

LSTM - Long Short-Term Neural Network

LSTM - Long Short-Term Neural Network

- É uma **Rede Neural Recorrente** (RNN).
- Possuem duas fontes de entrada: O **presente** e o **passado recente**.
- Possuem um **loop de feedback** conectado às suas decisões passadas.
- Estrutura dividida em **células**.



```
In [1]: import tensorflow as tf
        import numpy as np
        import matplotlib.pyplot as plt
        import matplotlib.pyplot as plt2
        import pandas as pd
        from pandas import datetime
        import math, time
        import itertools
        from sklearn import preprocessing
        import datetime
        from sklearn.metrics import mean_squared_error
        from math import sqrt
        from keras.models import Sequential
        from keras.layers.core import Dense, Dropout, Activation
        from keras.layers.recurrent import LSTM
        from keras.models import load_model
        import keras
        import pandas_datareader.data as web
        from keras.utils import plot_model
        import h5py
```

```
Using TensorFlow backend.
```

```
In [2]: seq_len = 22
        d = 0.2
        shape = [4, seq_len, 1] # feature, window, output
        neurons = [128, 128, 32, 1]
        epochs = 300
```

Normalizando os dados

In [6]:
```
df = raw_price_data
```

In [7]:
```
df = raw_price_data
df.drop(['Volume', 'Market Cap'], 1, inplace=True)

min_max_scaler = preprocessing.MinMaxScaler()
df['Open'] = min_max_scaler.fit_transform(df.Open.values.reshape(-1,1))
df['High'] = min_max_scaler.fit_transform(df.High.values.reshape(-1,1))
df['Low'] = min_max_scaler.fit_transform(df.Low.values.reshape(-1,1))
df['Close'] = min_max_scaler.fit_transform(df.Low.values.reshape(-1,1))
print(df.head())
plt.plot(df['High'], color='red', label='High')
plt.legend(loc='best')
plt.show()
```

In [23]:

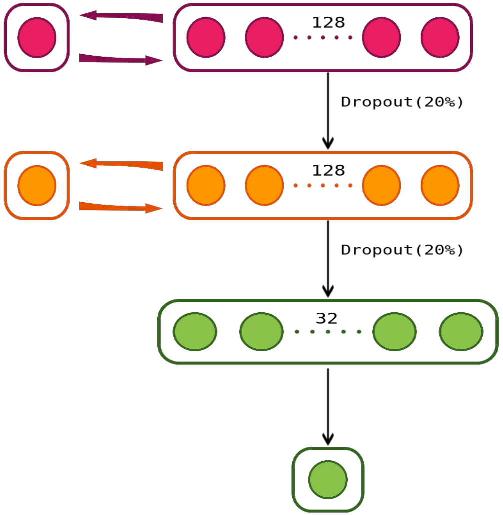
```
from sklearn.model_selection import train_test_split

training = df

x_train, x_test, y_train, y_test = train_test_split(
    training, training.High, train_size=0.85
)
print(len(x_train))
print(len(x_test))
print(len(y_train))
print(len(y_test))
```

```
1322
234
1322
234
```

```
In [10]: # Stack 3 LSTM layers on top of each other, making the model capable of learning higher-level temporal representatio
         def build_model(layers, neurons, d):
             model = Sequential()

             model.add(LSTM(neurons[0], input_shape=(layers[1], layers[0]), return_sequences=True))
             # Dropout consists in randomly setting a fraction rate of input units to 0 at each update during training time,
             model.add(Dropout(d))

             model.add(LSTM(neurons[1], input_shape=(layers[1], layers[0]), return_sequences=False))
             model.add(Dropout(d))

             model.add(Dense(neurons[2],kernel_initializer="uniform",activation='relu'))
             model.add(Dense(neurons[3],kernel_initializer="uniform",activation='linear'))
             model.compile(loss='mse',optimizer='adam', metrics=['accuracy'])
             model.summary()
             return model
```

```
In [12]: model = build_model(shape, neurons, d)
         # layers = [4, 22, 1]
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 22, 128)           68096
_________________________________________________________________
dropout_1 (Dropout)          (None, 22, 128)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 128)               131584
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                4128
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
=================================================================
Total params: 203,841
Trainable params: 203,841
Non-trainable params: 0
_________________________________________________________________
```



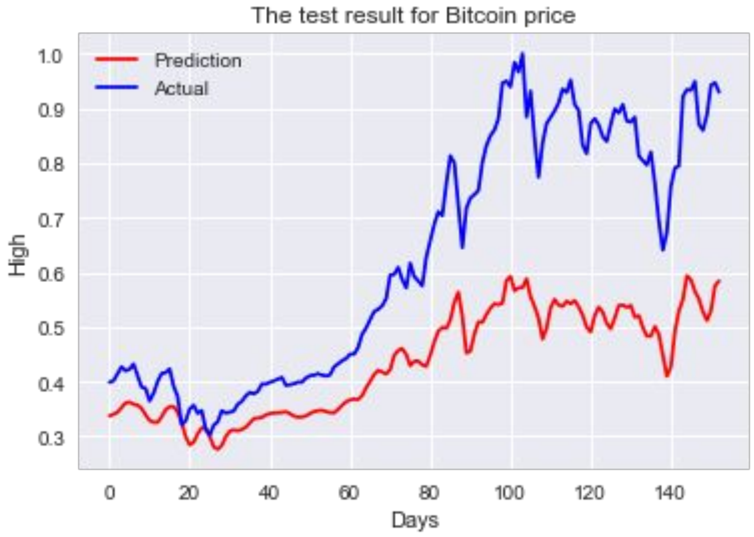
In [30]:

```python
model.fit(
    x_train,
    y_train,
    batch_size=512,
    epochs=epochs,
    validation_split=0.1,
    verbose=1)
```

```
Train on 1242 samples, validate on 138 samples
Epoch 1/300
1242/1242 [==============================] - 3s - loss: 0.0182 - acc: 8.0515e-04 - val_loss: 0.0712 - val_acc: 0.
0000e+00
Epoch 2/300
1242/1242 [==============================] - 2s - loss: 0.0154 - acc: 8.0515e-04 - val_loss: 0.0508 - val_acc: 0.
0000e+00
Epoch 3/300
1242/1242 [==============================] - 2s - loss: 0.0081 - acc: 8.0515e-04 - val_loss: 0.0031 - val_acc: 0.
0000e+00
Epoch 4/300
1242/1242 [==============================] - 3s - loss: 0.0041 - acc: 8.0515e-04 - val_loss: 0.0040 - val_acc: 0.
0000e+00
Epoch 5/300
1242/1242 [==============================] - 2s - loss: 0.0017 - acc: 8.0515e-04 - val_loss: 0.0189 - val_acc: 0.
0000e+00
Epoch 6/300
1242/1242 [==============================] - 2s - loss: 0.0030 - acc: 8.0515e-04 - val_loss: 0.0180 - val_acc: 0.
0000e+00
```

In [32]:
```python
p = model.predict(x_test)
```

In [46]:
```python
newp = denormalize(p)
newy_test = denormalize(y_test)
plt2.plot(newp, color='red', label='Prediction')
plt2.plot(newy_test,color='blue', label='Actual')
plt2.legend(loc='best')
plt2.title('The test result for {}'.format("Bitcoin price"))
plt2.xlabel('Days')
plt2.ylabel('High')
plt2.show()
```



PiSimo / **BitcoinForecast**

<> Code | Issues 2 | Pull requests 0 | Projects 0

Predict bitcoin price with deep learning

brandonrobertz / **BitcoinTradingAlgorithmToolkit**

Watch 22 | Star 101

<> Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights

A framework for logging, simulating, and analyzing prices of crypto currencies on various exchanges using technical analysis, fuzzy logic, and neural networks.

cbyn / **bitpredict**

<> Code | Pull requests 0 | Projects 0 | Insights

Machine learning for high frequency bitcoin price prediction

Link- / **bitcoin-manalysis**

<> Code | Issues 0 | Pull requests 0 | Projects 0

Bitcoin Data Analysis using Jupyter/pandas and matplotlib

I WANT YOU

Slides:

bit.ly/pyne-series

Twitter: https://twitter.com/rsarai007

Github: https://github.com/rsarai

Email: rebeca@vinta.com.br