En esta actividad vamos a analizar el gráfico del precio de diferentes criptomonedas con el objetivo de detectar tendencias. Antes de nada, vamos a ver los datos históricos expuestos en los datasets obtenidos de kaggle. Como modelo de referencia usaremos los datos del Bitcoin.

Lo primero que haremos será importar las bibliotecas que vamos a utilizar.

In [1]:
```
import requests
import pandas as pd
from datetime import datetime
import matplotlib.pyplot as plt
```

Ahora vamos a echar un vistazo a los datos que tenemos de Bitcoin

In [2]:
```
df = pd.read_csv("historicos/coin_Bitcoin.csv")
df
```

Out[2]:

	SNo	Name	Symbol	Date	High	Low	Open	Close	Volume	Marketcap
0	1	Bitcoin	BTC	2013-04-29 23:59:59	147.488007	134.000000	134.444000	144.539993	0.000000e+00	1.603769e+09
1	2	Bitcoin	BTC	2013-04-30 23:59:59	146.929993	134.050003	144.000000	139.000000	0.000000e+00	1.542813e+09
2	3	Bitcoin	BTC	2013-05-01 23:59:59	139.889999	107.720001	139.000000	116.989998	0.000000e+00	1.298955e+09
3	4	Bitcoin	BTC	2013-05-02 23:59:59	125.599998	92.281898	116.379997	105.209999	0.000000e+00	1.168517e+09
4	5	Bitcoin	BTC	2013-05-03 23:59:59	108.127998	79.099998	106.250000	97.750000	0.000000e+00	1.085995e+09
...	...	...	...	...	...	...	...	...	...	...
2986	2987	Bitcoin	BTC	2021-07-02 23:59:59	33939.588699	32770.680780	33549.600177	33897.048590	3.872897e+10	6.354508e+11
2987	2988	Bitcoin	BTC	2021-07-03 23:59:59	34909.259899	33402.696536	33854.421362	34668.548402	2.438396e+10	6.499397e+11
2988	2989	Bitcoin	BTC	2021-07-04 23:59:59	35937.567147	34396.477458	34665.564866	35287.779766	2.492431e+10	6.615748e+11
2989	2990	Bitcoin	BTC	2021-07-05 23:59:59	35284.344430	33213.661034	35284.344430	33746.002456	2.672155e+10	6.326962e+11
2990	2991	Bitcoin	BTC	2021-07-06 23:59:59	35038.536363	33599.916169	33723.509655	34235.193451	2.650126e+10	6.418992e+11

2991 rows × 10 columns

Aquí inmediatamente saltan a la vista dos problemas. El primero es que sólo hay datos hasta el 6 de julio de 2021, cuando el mercado ha cambiado enormemente en este último medio año. El segundo es que hay varias métricas que no vamos a utilizar.

Antes de modificar nada, vamos a ver de dónde podemos sacar los datos más recientes, lo cual es mucho más difícil de lo que parece. Tras investigar, hemos visto que la única forma de obtener los datos más recientes es haciendo peticiones a páginas que ofrecen este tipo de servicios. Estas páginas son conscientes del valor de estos datos, por ello sólo dan acceso completo a sus datos a aquellos usuarios que paguen una subscripción. De todas formas la versión gratuita nos da información sobre los últimos 200 días, y dado que desde el día actual hasta el último registro del dataset hay menos de 200 días, podemos usar el servicio gratuito sin problemas.

Para ver cómo lo podríamos juntar, vamos a ver en qué formato se encuentran estos datos haciendo una petición.

In [4]:
```
apiUrl = "https://api.pro.coinbase.com" # Enlace de la API
sym = "BTC-USD" # Símbolo del par de divisas cuya relación queremos obtener
end_date = datetime.now().date().isoformat()
start_date = "2021-07-07"
barsize = "86400" # Tiempo entre registros (en segundos)
parameters = {
    "start": start_date,
    "end": end_date,
    "granularity": barsize,
}
headers = {"content-type": "application/json"}
data = requests.get(f"{apiUrl}/products/{sym}/candles", params=parameters, headers=headers)
datos = data.json()
datos
```

Out[4]:

[[1641168000, 45737, 47583.33, 47299.06, 45792.19, 8626.12740493],
[1641081600, 46633.36, 47990, 47733.43, 47299.07, 6833.49845548],
[1640995200, 46205, 47967.12, 46211.24, 47733.43, 9463.6617111],
[1640908800, 45650, 48574.7, 47122.09, 46211.24, 19010.24134509],
[1640822400, 45938.44, 47926.15, 46471.24, 47122.08, 27413.65364976],
[1640736000, 46094.02, 48149.58, 47542.2, 46471.24, 20326.32142372],
[1640649600, 47300.23, 50720.35, 50720.35, 47543.09, 23116.75583707],
[1640563200, 50480, 52100, 50804.33, 50717.77, 11596.91468191],
[1640476800, 49460, 51295.33, 50428.31, 50801.79, 6863.39473781],
[1640390400, 50191.84, 51171.68, 50852.31, 50428.31, 5550.57153185],
[1640304000, 50445.55, 51878.6, 50842.06, 50851.38, 11690.45418179],
[1640217600, 48032.16, 51397.82, 48608.61, 50842.2, 16627.04627954],
[1640131200, 48450, 49595, 48914.7, 48608.62, 10753.4990677],
[1640044800, 46645.05, 49339.31, 46926.07, 48914.7, 15461.4105731],
[1639958400, 45568, 47548.93, 46687.2, 46926.07, 16039.38510407],
[1639872000, 46440.11, 48351.92, 46857.49, 46687.19, 9518.7071282],
[1639785600, 45515.72, 47368.73, 46159.87, 46859.46, 8170.85303229],
[1639699200, 45469.32, 48000, 47634.2, 46166.5, 18556.07900007],
[1639612800, 47524.66, 49459.39, 48878.83, 47634.22, 12731.23681745],
[1639526400, 46530, 49500, 48359.23, 48884.78, 21621.70983741],
[1639440000, 46300, 48686.91, 46727.89, 48367.43, 17107.0342879],
[1639353600, 45727.92, 50218.42, 50089.64, 46727.89, 20797.28834629],
[1639267200, 48665.43, 50828.13, 49424.47, 50089.64, 10411.08180306],
[1639180800, 46786.78, 49517.18, 47170.94, 49407.75, 11045.22374119],
[1639094400, 46900, 50148.49, 47568.43, 47170.94, 15689.5742182],
[1639008000, 47323.23, 50844.86, 50520.94, 47568.43, 15749.4243393],
[1638921600, 48650.02, 51250, 50625.24, 50519.68, 13138.99183299],
[1638835200, 50067.1, 51995, 50529.56, 50625.48, 14107.33216632],
[1638748800, 47200, 51105, 49484.21, 50529.56, 23082.30465624],
[1638662400, 47827, 49783, 49235.26, 49484.22, 21727.13821193],
[1638576000, 42333, 53876.09, 53633.02, 49241.12, 39023.32978526],
[1638489600, 51640, 57670.68, 56521.45, 53638.04, 19567.63113833],
[1638403200, 55845, 57423.69, 57226.51, 56521.46, 12969.21952648],
[1638316800, 56465.75, 59118.84, 56998.35, 57226.5, 16607.30872879],
[1638230400, 55910.33, 59249.77, 57838.06, 56987.97, 18038.78770838],
[1638144000, 56730, 58908.27, 57318.51, 57838.06, 13454.23335496],
[1638057600, 53327, 57490, 54759.04, 57317.17, 8222.42493442],
[1637971200, 53642.58, 55316.67, 53757.67, 54759.05, 7899.30249126],
[1637884800, 53533, 59194, 58994.72, 53757.67, 25946.30174683],
[1637798400, 57037.04, 59445.99, 57158.58, 58987.27, 10668.67680723],
[1637712000, 55875.01, 57607.97, 57562.65, 57162.66, 12509.25971944],
[1637625600, 55377, 57886.03, 56280.81, 57566.85, 14019.78167156],
[1637539200, 55641.03, 59526.51, 58671.22, 56280.81, 15889.22078935],
[1637452800, 58509.69, 60070, 59760.76, 58671.21, 6661.36635434],
[1637366400, 57423.35, 59900, 58122.15, 59760.76, 7584.65241653],
[1637280000, 55625, 58412.12, 56898.46, 58122.16, 18554.56064192],
[1637193600, 56514.13, 60977.21, 60351.51, 56898, 26749.21203012],
[1637107200, 58380, 60824.38, 60101.56, 60351.51, 20907.21545076],
[1637020800, 58638, 63624.6, 63621.05, 60107.98, 24910.15267457],
[1636934400, 63371, 66339.9, 65505.03, 63624.59, 10426.55522646],
[1636848000, 63596.9, 65525, 64400.01, 65505.02, 6081.57225975],
[1636761600, 63394.17, 64990, 64144.42, 64400.01, 5968.10354833],
[1636675200, 62300, 65477.06, 64807.73, 64147.9, 13658.626579],
[1636588800, 64133.78, 65600, 64912.2, 64807.59, 10259.51472415],
[1636502400, 62800, 69000, 66938.76, 64912.2, 19030.24517744],
[1636416000, 66261.79, 68568.85, 67554.13, 66944.66, 14996.62508818],
[1636329600, 63309.12, 67792.77, 63309.12, 67554.84, 17661.87297159],
[1636243200, 61397.86, 63327.96, 61539.3, 63309.13, 5596.02460734],
[1636156800, 60125, 61599.25, 61006.14, 61539.31, 6366.43859189],
[1636070400, 60777, 62638.59, 61444.49, 61006.15, 10598.79620286],
[1635984000, 60728.77, 63114.14, 62935.4, 61444.5, 11051.75853921],
[1635897600, 60070, 63568.36, 63266.5, 62935.41, 12951.87181247],
[1635811200, 60672.16, 64300, 60956.4, 63266.51, 13694.15392462],
[1635724800, 59500, 62500, 61346.17, 60949.54, 11724.34161877],
[1635638400, 60001, 62427.02, 61896.38, 61343.68, 8371.1698899],
[1635552000, 60725.01, 62379.2, 62287.95, 61892.41, 6793.59698552],
[1635465600, 60188.73, 62974, 60591.63, 62276.72, 15497.01101939],
[1635379200, 57653.88, 62477.47, 58462.73, 60591.65, 19354.49987509],
[1635292800, 58100, 61488, 60333.19, 58455.47, 18856.68565412],
[1635206400, 59837.07, 63290.49, 63081.96, 60337.62, 12181.48144997],
[1635120000, 60653.5, 63726.58, 60847.91, 63083.54, 10496.58058683],
[1635033600, 59522.89, 61491.96, 61300.02, 60854.48, 8573.72719199],
[1634947200, 59650, 61750, 60682.88, 61300.01, 6708.15013802],
[1634860800, 60000, 63750, 62204.01, 60687.64, 17217.70765017],
[1634774400, 62050, 66650.85, 66021.26, 62204.02, 23474.39133313],
[1634688000, 63525, 66999, 64303.14, 66026.54, 19194.45410878],
[1634601600, 61333, 64500, 62042.4, 64303.14, 17380.30486176],
[1634515200, 59887.74, 62675.19, 61530.07, 62042.41, 19798.1489024],
[1634428800, 58943.83, 61744.17, 60867.05, 61527.11, 9153.32273471],
[1634342400, 60139.01, 62350, 61690.32, 60877.42, 10161.25819825],
[1634256000, 56867.11, 62910, 57359.51, 61695.39, 27105.19764492],
[1634169600, 56832.32, 58520.71, 57367.32, 57359.51, 12198.02457938],
[1634083200, 54236.09, 57771.33, 56005.1, 57367.32, 14691.01338574],
[1633996800, 53873.78, 57674.55, 57485.97, 56005.1, 15457.00896085],
[1633910400, 54411.54, 57833.23, 54683.09, 57487.44, 13422.42081815],
[1633824000, 54112.95, 56545.24, 54963.29, 54690.53, 9370.1912382],
[1633737600, 53675, 55500, 53965.18, 54962.29, 7123.05512404],
[1633651200, 53634.41, 56113, 53805.46, 53963.82, 12416.9019906],
[1633564800, 53379, 55356.85, 55346.95, 53797.82, 15820.83471633],
[1633478400, 50416.01, 55757.11, 51499.77, 55339.48, 25869.64925998],
[1633392000, 49057.18, 51906.23, 49244.13, 51493.99, 17828.59726658],
[1633305600, 46916.7, 49505, 48233.99, 49245.54, 13938.64182971],
[1633219200, 47120.11, 49300, 47663.74, 48233.99, 7989.25925892],
[1633132800, 47451, 48359.33, 48165.76, 47657.69, 7262.63442143],
[1633046400, 43287.44, 48500, 43828.89, 48165.76, 20095.33155568],
[1632960000, 41409.67, 44110.17, 41519.11, 43824.43, 12835.9838101],
[1632873600, 40750.12, 42638.99, 41030.19, 41522.16, 10303.30152312],
[1632787200, 40897.7, 42780, 42166.31, 41026.07, 10491.81363045],
[1632700800, 42111, 44366.96, 43172.17, 42171.76, 10094.77099806],
[1632614400, 40803, 43937, 42705.51, 43178.02, 9971.64545213],
[1632528000, 41675, 42998.76, 42848.44, 42705.51, 7383.32390261],
[1632441600, 40683.29, 45200, 44896.55, 42848.92, 22451.57711462],
[1632355200, 43096.73, 45000, 43575.19, 44897.59, 13446.87468684],
[1632268800, 40570.42, 44035.44, 40736.81, 43575.1, 19462.53865157],
[1632182400, 39600, 43639.88, 43007.69, 40719.6, 27275.33818249],
[1632096000, 42500, 47358.92, 47253.71, 43012.97, 27706.67636162],
[1632009600, 46850.87, 48379.19, 48309.86, 47255.92, 7086.34626236],
[1631923200, 47050, 48825.62, 47306.87, 48314.56, 7491.03449237],
[1631836800, 46752.67, 48176.64, 47753.15, 47303.5, 9088.56956453],
[1631750400, 47020.82, 48499.99, 48144.03, 47753.16, 11739.27966629],
[1631664000, 46705, 48475, 47127.81, 48148.12, 12139.84206628],
[1631577600, 44679.92, 47274.92, 44953.23, 47127.22, 12644.16435358],
[1631491200, 43465, 46900, 46027.8, 44947.72, 15015.30289712],
[1631404800, 44754.31, 46462.98, 45173.66, 46024.23, 6499.85821749],
[1631318400, 44730.29, 45989.94, 44850.37, 45171.83, 6207.38337004],
[1631232000, 44140.48, 47040.76, 46396.26, 44851.45, 14905.54795341],
[1631145600, 45511.82, 47400, 46059.94, 46400, 13594.64633403],
[1631059200, 44423.49, 47381.47, 46894.49, 46060.36, 19624.55093117],
[1630972800, 42830.77, 52944.96, 52698.8, 46894.5, 29022.815351],
[1630886400, 51020.08, 52802.03, 51789.17, 52698.81, 9571.75471136],
[1630800000, 49500, 51907.08, 49944.89, 51789.17, 7743.93472308],
[1630713600, 49400, 50558.75, 50025, 49942.98, 7595.48816499],
[1630627200, 48349.66, 51064.44, 49279.38, 50025, 14577.22621879],
[1630540800, 48620.73, 50392.19, 48834.27, 49279.37, 14120.3826407],
[1630454400, 46537.62, 49125, 47110.33, 48839.86, 13010.83839478],
[1630368000, 46708.93, 48259.68, 46996.8, 47112.5, 12919.43970212],
[1630281600, 46866, 48906.51, 48802.58, 46993.71, 12249.23741495],
[1630195200, 47801, 49667.1, 48942.89, 48802.58, 6935.09511006],
[1630108800, 48370, 49309.83, 49081.08, 48942.94, 4895.70191319],
[1630022400, 46361.01, 49185.12, 46851.47, 49077.57, 8924.12247655],
[1629936000, 46315.48, 49365.1, 48991.57, 46851.47, 11651.34373853],
[1629849600, 47122.02, 49277, 47682.45, 48987.33, 9289.23177109],
[1629763200, 47600, 49875.9, 49504.53, 47682.45, 12068.33823315],
[1629676800, 49012.84, 50505, 49284.38, 49506.5, 10824.26838095],
[1629590400, 48102.89, 49526.13, 48870.21, 49284.63, 6243.39643384],
[1629504000, 48285.28, 49821.92, 49349.99, 48867.02, 9574.83615164],
[1629417600, 46644.95, 49400, 46769.02, 49342.4, 13184.67048641],
[1629331200, 43955, 47088.08, 44709.1, 46765.87, 11938.12185013],
[1629244800, 44216.47, 46035.27, 44675.34, 44707.98, 10327.49887365],
[1629158400, 44405.39, 47174.62, 45903.22, 44671.58, 12756.14779945],
[1629072000, 45676.91, 48076.52, 47014.38, 45902.46, 10467.66031475],
[1628985600, 45525.19, 47400, 47109.64, 47014.49, 7745.70200802],
[1628899200, 46034.93, 48176.52, 47833.64, 47109.65, 8130.36769388],
[1628812800, 44249.64, 47953.9, 44422.15, 47833.64, 11794.94196257],
[1628726400, 43714.36, 46223.95, 45553.1, 44422.14, 13169.78176994],
[1628640000, 45350.74, 46781.09, 45601.82, 45553.49, 11628.32043954],
[1628553600, 44650.27, 46718.44, 46280, 45595.66, 13756.50066875],
[1628467200, 42821.3, 46497.42, 43829.15, 46285.48, 19464.72493503],
[1628380800, 43131.97, 45363.83, 44628.11, 43829.14, 18712.27158173],
[1628294400, 42457.27, 44750, 42869.59, 44637.34, 19634.79475606],
[1628208000, 39876.74, 43399.26, 40888.73, 42869.58, 20425.1145444],
[1628121600, 37300, 41447.89, 39718.9, 40888.74, 28507.85532682],
[1628035200, 37509, 39965.1, 38189.4, 39722.41, 15327.67624669],
[1627948800, 37627.21, 39785.3, 39149.94, 38191.44, 15094.81739624],
[1627862400, 38690, 40459.68, 39859.18, 39149.59, 12379.41216528],
[1627776000, 39426.92, 42605.64, 41499, 39865.41, 10617.67293456],
[1627689600, 41064.11, 42414.61, 42233.66, 41495.01, 9351.94288576],
[1627603200, 38342, 42320.57, 40039.44, 42237.95, 15063.2344059],
[1627516800, 39229.47, 40644.85, 40035.1, 40039.44, 11022.42512817],
[1627430400, 38803.34, 40925.56, 39475.99, 40035.1, 27198.33358111],
[1627344000, 36413.26, 39550, 37262.76, 39470.48, 24709.10440908],
[1627257600, 35251.54, 40593.93, 35428.25, 37262.77, 40354.54555841],
[1627171200, 33888.89, 35451.04, 34291.67, 35428.26, 9434.63002297],
[1627084800, 33424.7, 34525, 33647.27, 34283.01, 9864.75243703],
[1626998400, 32000, 33647.27, 32287.75, 33647.26, 9004.84262627],
[1626912000, 31729.68, 32611.84, 32152.68, 32287.74, 7704.6404628],
[1626825600, 29501.02, 32825, 29796.15, 32135.19, 15598.47830878],
[1626739200, 29301.56, 31052.65, 30842.04, 29796.16, 18114.15286104],
[1626652800, 30429.99, 31887.82, 31786.37, 30842.03, 11027.00528966],
[1626566400, 31125.6, 32450, 31533.9, 31788.25, 5006.4831051],
[1626480000, 31179.01, 31949.99, 31389.57, 31533.91, 5695.57411588],
[1626393600, 31025.42, 32259.16, 31872.49, 31388.06, 10006.47793721],
[1626307200, 31064.77, 33187.6, 32815.75, 31868.68, 11273.61174025],
[1626220800, 31600, 33125.55, 32734.14, 32816.39, 10355.3186961],
[1626134400, 32201.12, 33337.61, 33077.43, 32734.14, 9478.04218884],
[1626048000, 32665, 34670, 34259.22, 33091.1, 10145.24046995],
[1625961600, 33333.33, 34607.37, 33515.35, 34259.23, 7023.06500489],
[1625875200, 33027.83, 34267.14, 33824.25, 33515.35, 4906.24582215],
[1625788800, 32255.24, 34100, 32875.95, 33824.26, 8320.90649257],
[1625702400, 32111, 33934.62, 33879.5, 32875.95, 14147.35325248],
[1625616000, 33770.01, 35077.46, 34225.72, 33878.56, 9404.8979867]]

Vemos que obtenemos los datos en forma de array de arrays. Por cómo están entrelazados, vemos que aparte de la timestamp tenemos el precio mínimo, máximo, de apertura y de clausura diarios respectivamente. Además, al analizar los timestamps vemos que los datos están al revés de como querríamos verlos (va de día actual a 7 de julio).

Vamos a modificar los datasets originales para que tenga los datos que nos interesan, luego uniremos el resultado junto con estos resultados que obtenemos de la petición.

In [5]:
```
def modify_lines(line): # Modificamos cada línea del cav para dejar los datos que
    data = line.split(',') # vayan a jugar un papel importante en nuestro análisis
    return "{},{},{:.2f},{:.2f},{:.2f},{:.2f}\n".format(data[2], data[3][:10], float(data[4]), float(data[5]), float(data[6]), float(data[7]))

def get_lines(file): # Obtenemos cada línea del csv como array, luego las modificamos
    with open(file, "r") as coin:
        lines = coin.readlines()
        return ["Symbol,Date,High,Low,Open,Close\n"] + [modify_lines(line) for line in lines[1::]]


with open("historicos/bitcoin_limpio.csv", "w") as coin: # Creamos un nuevo archivo con nuestras modificaciones
    coin.writelines(get_lines("historicos/coin_Bitcoin.csv"))
```

Veamos qué aspecto tiene nuestro dataset modificado

In [6]:
```
cleaned_df = pd.read_csv("historicos/bitcoin_limpio.csv")
cleaned_df
```

Out[6]:

	Symbol	Date	High	Low	Open	Close
0	BTC	2013-04-29	147.49	134.00	134.44	144.54
1	BTC	2013-04-30	146.93	134.05	144.00	139.00
2	BTC	2013-05-01	139.89	107.72	139.00	116.99
3	BTC	2013-05-02	125.60	92.28	116.38	105.21
4	BTC	2013-05-03	108.13	79.10	106.25	97.75
...	...	...	...	...	...	...
2986	BTC	2021-07-02	33939.59	32770.68	33549.60	33897.05
2987	BTC	2021-07-03	34909.26	33402.70	33854.42	34668.55
2988	BTC	2021-07-04	35937.57	34396.48	34665.56	35287.78
2989	BTC	2021-07-05	35284.34	33213.66	35284.34	33746.00
2990	BTC	2021-07-06	35038.54	33599.92	33723.51	34235.19

2991 rows × 6 columns

Perfecto, mucho más limpio y entendible. Ahora vamos a modificar los resultados la API para poder añadirlos. Vamos a meterlos también en una función para que realizar estas operaciones más cómodamente en el resto de datasets.

In [7]:
```
apiUrl = "https://api.pro.coinbase.com"
sym = "BTC-USD"
start_date = "2021-07-07"

def get_prices(sym, end_date, start_date):
    barsize = "86400" #  Tiempo en segundos entre registros
    parameters = {
        "start": start_date,
        "end": end_date,
        "granularity": barsize,
    }
    headers = {"content-type": "application/json"}
    data = requests.get(f"{apiUrl}/products/{sym}/candles", params=parameters, headers=headers)
    datos = data.json()[::-1] # Damos la vuelta para obtener primero los más antiguos
    return datos

precios = get_prices(sym, end_date=end_date, start_date=start_date)

fechas = pd.date_range(start=start_date, end=end_date).to_list()
fechas = [time.strftime("%Y-%m-%d") for time in fechas] # Obtenemos fechas de cada registro

todo_junto = [f"{sym[:3]},{fechas[i]},{precios[i][2]},{precios[i][1]},{precios[i][3]},{precios[i][4]}\n" for i in range(len(precios))]
# Especificamos los campos para el nuevo csv. Los elementos 1 y 2 han sido invertidos para
# tener el mismo formato que el csv original.

with open("historicos/bitcoin_limpio.csv", "a") as coin: # Completamos nuestro csv limpiado
    coin.writelines(todo_junto)
```

Ahora comprobamos que en nuestro dataframe se han escrito los datos correctamente

In [8]:
```
final_df = pd.read_csv("historicos/bitcoin_limpio.csv")
final_df
```

Out[8]:

	Symbol	Date	High	Low	Open	Close
0	BTC	2013-04-29	147.49	134.00	134.44	144.54
1	BTC	2013-04-30	146.93	134.05	144.00	139.00
2	BTC	2013-05-01	139.89	107.72	139.00	116.99
3	BTC	2013-05-02	125.60	92.28	116.38	105.21
4	BTC	2013-05-03	108.13	79.10	106.25	97.75
...	...	...	...	...	...	...
3167	BTC	2021-12-30	47926.15	45938.44	46471.24	47122.08
3168	BTC	2021-12-31	48574.70	45650.00	47122.09	46211.24
3169	BTC	2022-01-01	47967.12	46205.00	46211.24	47733.43
3170	BTC	2022-01-02	47990.00	46633.36	47733.43	47299.07
3171	BTC	2022-01-03	47583.33	45700.00	47299.06	46022.29

3172 rows × 6 columns

Todo pinta fantástico, es el momento de añadirle nuevos campos

In [9]:
```
final_df['variacion_diaria'] = final_df.apply(lambda row: (row.Close - row.Open) / row.Open * 100, axis=1)
# Añadimos una columna que nos indique cuánto ha cambiado porcentualmente el precio de la divisa cada día
final_df['promedio_movil_200_semanas'] = final_df['Open'].rolling(window = 1400).mean()
# Añadimos el promedio medio móvil de 200 semanas
final_df
```

Out[9]:

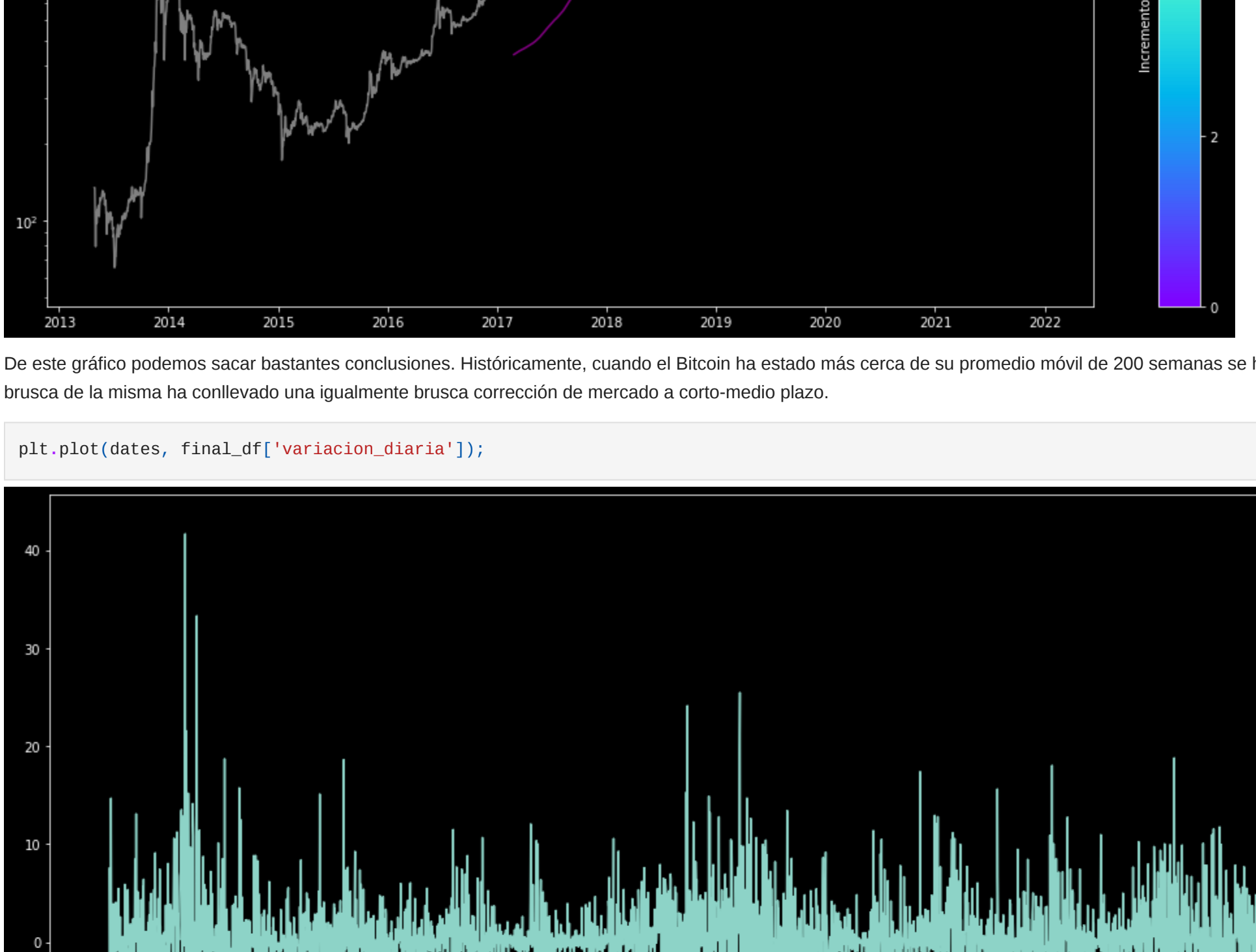
	Symbol	Date	High	Low	Open	Close	variacion_diaria	promedio_movil_200_semanas
0	BTC	2013-04-29	147.49	134.00	134.44	144.54	7.512645	NaN
1	BTC	2013-04-30	146.93	134.05	144.00	139.00	-3.472222	NaN
2	BTC	2013-05-01	139.89	107.72	139.00	116.99	-15.834532	NaN
3	BTC	2013-05-02	125.60	92.28	116.38	105.21	-9.597869	NaN
4	BTC	2013-05-03	108.13	79.10	106.25	97.75	-8.000000	NaN
...	...	...	...	...	...	...	...	...
3167	BTC	2021-12-30	47926.15	45938.44	46471.24	47122.08	1.400522	18633.023864
3168	BTC	2021-12-31	48574.70	45650.00	47122.09	46211.24	-1.932958	18658.841500
3169	BTC	2022-01-01	47967.12	46205.00	46211.24	47733.43	3.293982	18683.919600
3170	BTC	2022-01-02	47990.00	46633.36	47733.43	47299.07	-0.909970	18709.802479
3171	BTC	2022-01-03	47583.33	45700.00	47299.06	46022.29	-2.699356	18735.350093

3172 rows × 8 columns

Estos dos datos nos dan información valiosísima sobre la evolución del precio de bitcoin a lo largo de los años. Podemos visualizarlo en un gráfico para ver el gran valor de estos campos.

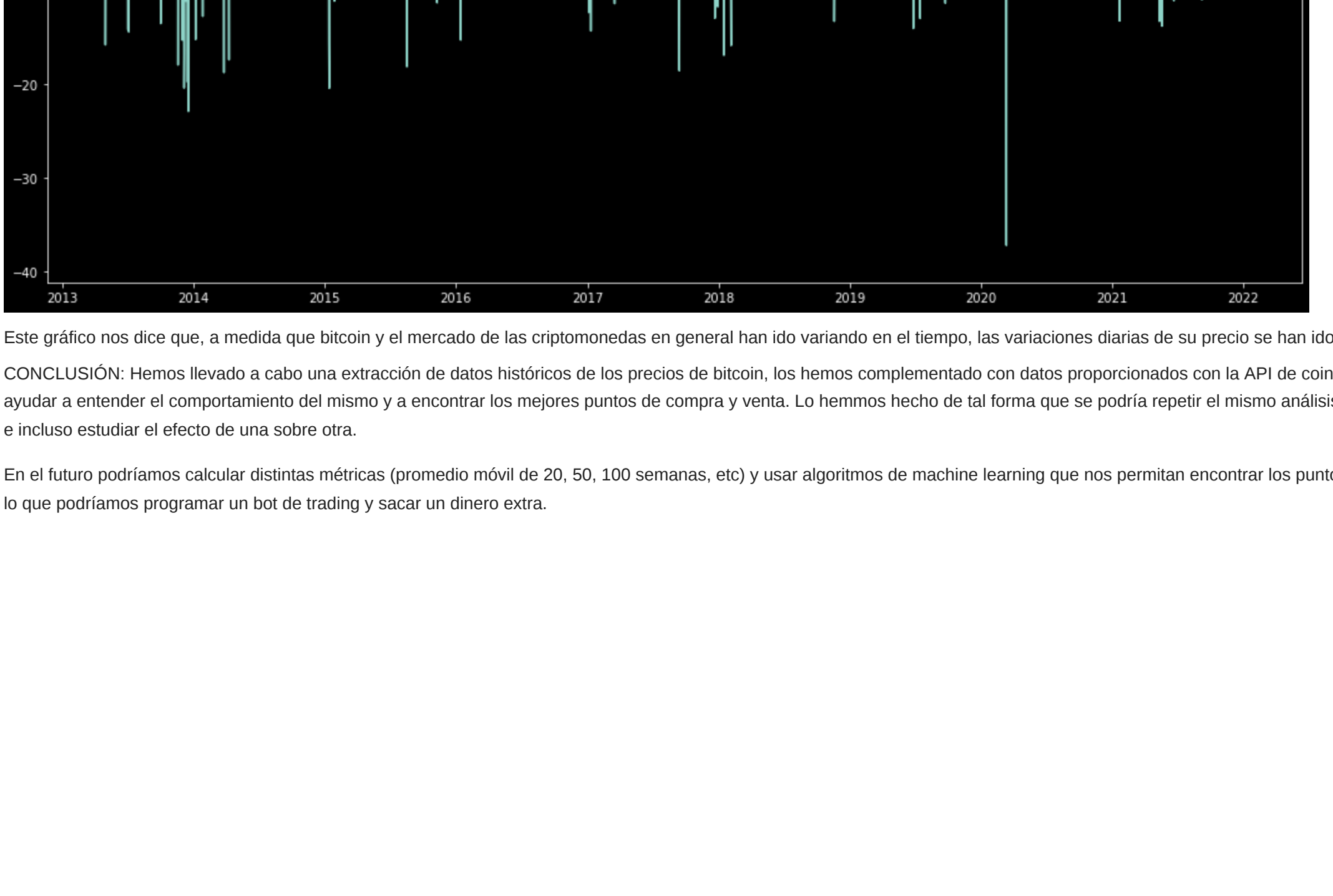
In [12]:
```
plt.rcParams['figure.figsize'] = [18, 12]
plt.style.use("dark_background")
mensual = final_df[::30]
distancia = mensual['promedio_movil_200_semanas'].pct_change() * 100
dates = pd.to_datetime(final_df['Date'])
plt.semilogy(dates, final_df['Low'], color="gray", zorder=1);
plt.semilogy(dates, final_df['promedio_movil_200_semanas'], color="purple", zorder=2);
plt.scatter(mensual['Date'], mensual['Open'], c=distancia, cmap='rainbow', vmin=0, vmax=10, zorder=3)

cbar = plt.colorbar()
cbar.set_label("Incremento mensual del promedio móvil 200 semanas (%)")
cbar.ax.yaxis.set_label_position("left")
```



De este gráfico podemos sacar bastantes conclusiones. Históricamente, cuando el Bitcoin ha estado más cerca de su promedio móvil de 200 semanas se han dado las mejores oportunidades de compra, mientras que una variación brusca de la misma ha conllevado una igualmente brusca corrección de mercado a corto-medio plazo.

In [15]:
```
plt.plot(dates, final_df['variacion_diaria']);
```



Este gráfico nos dice que, a medida que bitcoin y el mercado de las criptomonedas en general han ido variando en el tiempo, las variaciones diarias de su precio se han ido disminuyendo.

CONCLUSIÓN: Hemos llevado a cabo una extracción de datos históricos de los precios de bitcoin, los hemos complementado con datos proporcionados con la API de coinbase y hemos calculado distintas métricas que nos pueden ayudar a entender el comportamiento del mismo y a encontrar los mejores puntos de compra y venta. Lo hemmos hecho de tal forma que se podría repetir el mismo análisis fácilmente para cuelquiera de las criptomonedas propuestas, e incluso estudiar el efecto de una sobre otra.

En el futuro podríamos calcular distintas métricas (promedio móvil de 20, 50, 100 semanas, etc) y usar algoritmos de machine learning que nos permitan encontrar los puntos idóneos de compra y venta de cualquier criptomoneda, con lo que podríamos programar un bot de trading y sacar un dinero extra.