MÁSTER BIG DATA AND BUSINESS ANALYTICS

TRABAJO FIN DE MÁSTER

STOCK PRICE PREDICTION

Autores:

Gary Martin

Levi Vilchez

Javier Jimenez

Youssef Ouabi

Sebastian Esponda

Tutores:

Carlos Ortega

Santiago Mota

1. INTRODUCCIÓN

Los mercados financieros mueven millones de euros diarios. En los últimos años, sumado con la pandemia y los confinamientos ha aumentado significativamente el interés de las personas, especialmente de los jóvenes en los mercados de valores.

Una gran cantidad de inversores tanto pequeños como grandes, invierten de manera descontrolada su dinero por el ansia de conseguir un retorno elevado de su inversión.

Hoy en día invertir en bolsa es cada vez más fácil, ya que se puede hacer desde cualquier ordenador o a través de una aplicación con el móvil.

Adicionalmente, la rápida expansión de los brokers online, o corredores de bolsa en línea los cuales han abierto sus puertas a los inversionistas más inexpertos con cobros muy bajos de comisiones han aumentado la tendencia a invertir.

Así como ha aumentado el número de inversiones en bolsa, también ha aumentado el número de fracasos. Esto se debe a la falta de experiencia, pues con solo leer un poco y escuchar consejos de influencer muchas personas creen que están listos para invertir en bolsa. (Barria, 2021)

Pues si bien invertir en bolsa puede ayudar a que nuestros ahorros crezcan ya que la rentabilidad suele ser superior a lo que ofrecen las entidades bancarias, siempre se corre riesgo ya que no se puede predecir la rentabilidad de las acciones, es decir que aunque estas aumenten su valor durante algunos años, esto puede cambiar repentinamente debido a diferentes factores y si no se toman decisiones a tiempo se puede incluso llegar a perder todo.

En la actualidad, existe una forma más eficiente de invertir en bolsa, con ayuda de las nuevas tecnologías como el Big data, inteligencia artificial, machine learning y técnicas de Deep learning es posible predecir el comportamiento de los activos. Estas tecnologías nos ayudan a la toma de decisiones, proporcionando nuevas perspectivas que enriquecen dicho análisis.

Algunas de las técnicas más usuales son: procesamiento del lenguaje natural mediante lectura computacional (machine reading), reconocimiento de patrones (machine learning) y medición de la actividad de búsqueda por Internet para predecir los cambios en los volúmenes de ventas, entre otras. (BBVA, s.f)

En este trabajo se han utilizado distintas técnicas de Machine learning para intentar predecir cuándo es buen momento para invertir en bolsa. Se tomaron XXXXN empresas para ver el comportamiento de sus acciones a través del tiempo de manera que permita a las personas, aun a las que tienen poco o nulo conocimiento de inversion bursatil saber cuando es el mejor momento para comprar o vender acciones

2. OBJETIVOS

Obtener un modelo de machine learning que nos permita predecir el comportamiento de los stocks y nos ayude a tomar decisiones al momento de invertir

2.1. Objetivos específicos:

- Seleccionar el mejor modelo para la predicción de stocks.
- Crear un dashboard con los comportamientos y predicciones de los stocks para un período de () que se actualice cada (cierto tiempo)
- Implementación del modelo

3. PAQUETES UTILIZADOS

Para obtener los datos financieros de distintas empresas, hemos utilizado la librería **yfinance**, esta popular librería de open data permite acceder a los datos financieros disponibles en Yahoo Finance.

Yahoo Finance cuenta con un gran rango de datos de mercado sobre stocks, bonos, diferentes monedas y criptomonedas. Además ofrece noticias y análisis de los mercados.(Blad, 2020)

También fueron usadas las siguientes librerías a lo largo de este trabajo para creación de modelos y gráficos:

Plotly: es una biblioteca de dibujo interactiva basada en navegador, su función principal es dibujar gráficos interactivos en línea. El cuadro dibujado es realmente agradable a la vista y es muy fácil de utilizar. (Programador Clic, s.f)

Pycaret: Es una librería open source y low code para crear modelos de machine learning, en los últimos años ha ganado gran popularidad puesto que es muy fácil de usar, requiere poco código y se pueden hacer muchos modelos en poco tiempo.

Se podría decir que es una librería como H20 de AutoML que permite generar modelos prácticamente de manera semi automática.

Así mismo, también ofrece análisis exploratorio y despliegue de modelos mediante una API; también permite generar modelos de series temporales y forecast. (PyCaret, s.f)

Prophet Facebook: Es una librería de código abierto diseñada para realizar forecasting de conjuntos de datos de series temporales univariantes. Es fácil de usar y está diseñada para encontrar automáticamente un buen conjunto de hiperparámetros para el modelo en un esfuerzo por hacer previsiones hábiles para datos con tendencias y estructura estacional por defecto.

Está basado en Pytorch paquete de Python empleado para Deep Learning y no requiere gran cantidad de código. Los datos son normalizados automáticamente y se puede graficar predicciones futuras y pasadas para ver una comparativa de cómo funciona el modelo. (Letham, 2017)

NeuralProphet: Al igual que Prophet Facebook, esta es una librería de código abierto que utiliza el descenso de gradiente de PyTorch para la optimización, lo que hace que el modelado sea mucho más rápido. También es un modelo fácil al no requerir de gran cantidad de código. El forecasting utilizando la red neuronal Prophet es un modelo híbrido (Lineal - Red Neuronal).

De la misma forma que Prophet Facebook este modelo es que los datos son normalizados automáticamente y en cuanto a la visualización, se puede graficar predicciones futuras y pasadas para ver una comparativa de cómo funciona el modelo. (Neural Prophet, s.f.)

Keras: es una biblioteca de código abierto escrita en Python. Su objetivo es acelerar la creación de redes neuronales: para ello, Keras no funciona como un framework independiente, sino como una interfaz de uso intuitivo (API) que permite acceder a varios frameworks de aprendizaje automático y desarrollarlos. Entre los frameworks compatibles con Keras, se incluyen Theano, Microsoft Cognitive Toolkit (anteriormente CNTK) y TensorFlow. (Digital Guide IONOS, s.f)

3.1. ¿ Por qué Forecasting de series temporales?

La predicción de **series temporales** son datos recogidos del mismo tema en diferentes momentos temporales como el PIB de un país por año, el precio de las acciones de una empresa en particular durante un período de tiempo, o su propio latido del corazón registrado en cada segundo, de hecho, cualquier cosa que pueda capturar continuamente en diferentes intervalos de tiempo es un dato de serie temporal. Consiste en recopilar datos históricos, prepararlos para que los algoritmos los consuman (el algoritmo es, sencillamente, la matemática que va detrás de la escena), y luego predecir los valores futuros basándose en los patrones aprendidos de los datos históricos.

¿Se le ocurre alguna razón por la que las empresas o cualquier otra persona estarían interesadas en predecir los valores futuros de cualquier serie temporal (PIB, ventas mensuales, inventarios, desempleo, temperaturas globales, etc.)? Permítame darle una perspectiva empresarial:

Un minorista puede estar interesado en predecir las ventas futuras a nivel de SKU para planificar y presupuestar.

Un pequeño comerciante puede estar interesado en predecir las ventas por tienda, para poder programar los recursos adecuados (más personal durante los periodos de mayor actividad y viceversa).

Un gigante del software como Google puede estar interesado en conocer la hora más concurrida del día o el día más concurrido de la semana para poder programar los recursos del servidor en consecuencia.

El ministerio de Sanidad puede estar interesado en saber el número acumulado de vacunas COVID administradas para conocer el punto de consolidación en el que se espera que se produzca la inmunidad de rebaño.

4. EXPLORATORY DATA ANALYSIS

4.1 Inspección inicial de los datos (información, tamaño, descripción).

4.2 Exploratory Data Analysis (gráficos, correlaciones..).

4.3 Análisis financiero utilizando funciones de Yfinance (inversores de la empresa, dividendos, cashflow).

El primer paso será la instalación de las librerías “yfinance” y “plotly”:

```
!pip install yfinance
!pip install plotly
```

Además se procede a la carga de librerías conocidas como son “pandas” o “numpy” así como librerías específicas para la realización de este estudio financiero:

```
import yfinance as yf
import datetime as dt
# Data manipulation
import numpy as np
import pandas as pd

# Data viz
import plotly as px
import matplotlib.pyplot as plt
import seaborn as sns
```

En una primera instancia se han obtenido una serie de valores de Google. Se ha escogido como periodo el máximo número de observaciones y se han cargado los valores históricos (datos desde el 2004 hasta 2022, en un intervalo diario).

```
google = yf.download(tickers='GOOG', period='max', interval='1d')
```

4.1. Inspección inicial de los datos

El dataset cargado consta de 7 columnas, que son las siguientes:

Open: Hace referencia al precio de apertura.
High :El precio más alto de esa hora.
Low: El precio más bajo de esa hora
Close: El precio de cierre.
Adj Close: Precio ajustado
Volume: El volumen de transacciones en esa hora

Se consulta ahora el tamaño del dataset:

```
# Tamaño del dataset
tamaño = print('El conjunto de datos tiene', google.shape[0] ,'columnas y', google.shape[1], 'filas')
```

```
El conjunto de datos tiene 4406 columnas y 6 filas
```

Utilizando el comando “describe” se obtienen las principales características del dataset cargado:

	Open	High	Low	Close	Adj Close	Volume
count	4406.000000	4406.000000	4406.000000	4406.000000	4406.000000	4.406000e+03
mean	678.482003	685.002774	671.905314	678.615621	678.615621	6.419089e+06
std	633.583756	639.605449	627.739555	633.814192	633.814192	7.785755e+06
min	49.409801	50.680038	47.800831	49.818268	49.818268	7.922000e+03
25%	246.762573	249.057720	243.056469	246.585732	246.585732	1.558950e+06
50%	429.338242	431.512589	425.410446	428.414200	428.414200	3.731635e+06
75%	973.855011	981.042511	966.060013	973.227509	973.227509	8.113849e+06
max	3037.270020	3042.000000	2997.750000	3014.179932	3014.179932	8.254163e+07

En esta tabla pueden observarse valores como el precio medio (apertura, el más alto, el de cierre…), el precio máximo y mínimo histórico.

Centrando el estudio, por ejemplo, en las últimas 5 observaciones:

```
# Veamos las 5 primeras observaciones
google.head()
```

Date	Open	High	Low	Close	Adj Close	Volume
2004-08-19	49.813290	51.835709	47.800831	49.982655	49.982655	44871361
2004-08-20	50.316402	54.336334	50.062355	53.952770	53.952770	22942874
2004-08-23	55.168217	56.528118	54.321388	54.495735	54.495735	18342897
2004-08-24	55.412300	55.591629	51.591621	52.239197	52.239197	15319808
2004-08-25	52.284027	53.798351	51.746044	52.802086	52.802086	9232276

Con el objetivo de comprobar si los datos están limpios se procede a consultar la existencia de valores nulos:

```
# Comprobemos si existen valores nulos
total = google.isnull().sum().sort_values(ascending=False)

percent_1 = google.isnull().sum()/google.isnull().count()*100

percent_2 = (round(percent_1, 1)).sort_values(ascending=False)

missing_data = pd.concat([total, percent_2], axis=1, keys=['Total', '%'])

missing_data.head(15)
```

	Total	%
Volume	0	0.0
Adj Close	0	0.0
Close	0	0.0
Low	0	0.0
High	0	0.0
Open	0	0.0

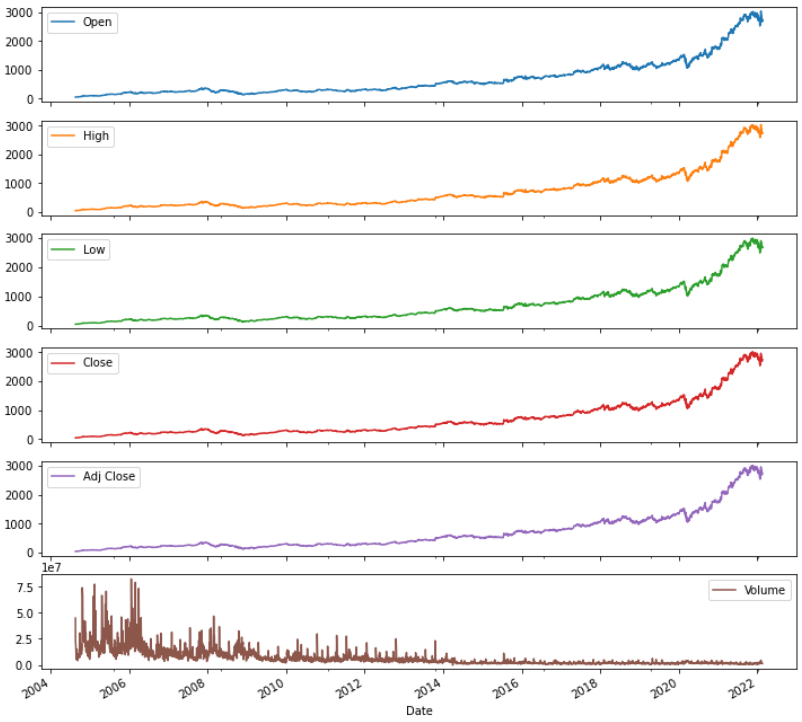
Así como de la existencia de valores duplicados:

```
# Así mismo es conveniente saber si hay observaciones duplicadas
sum(google.duplicated())
```

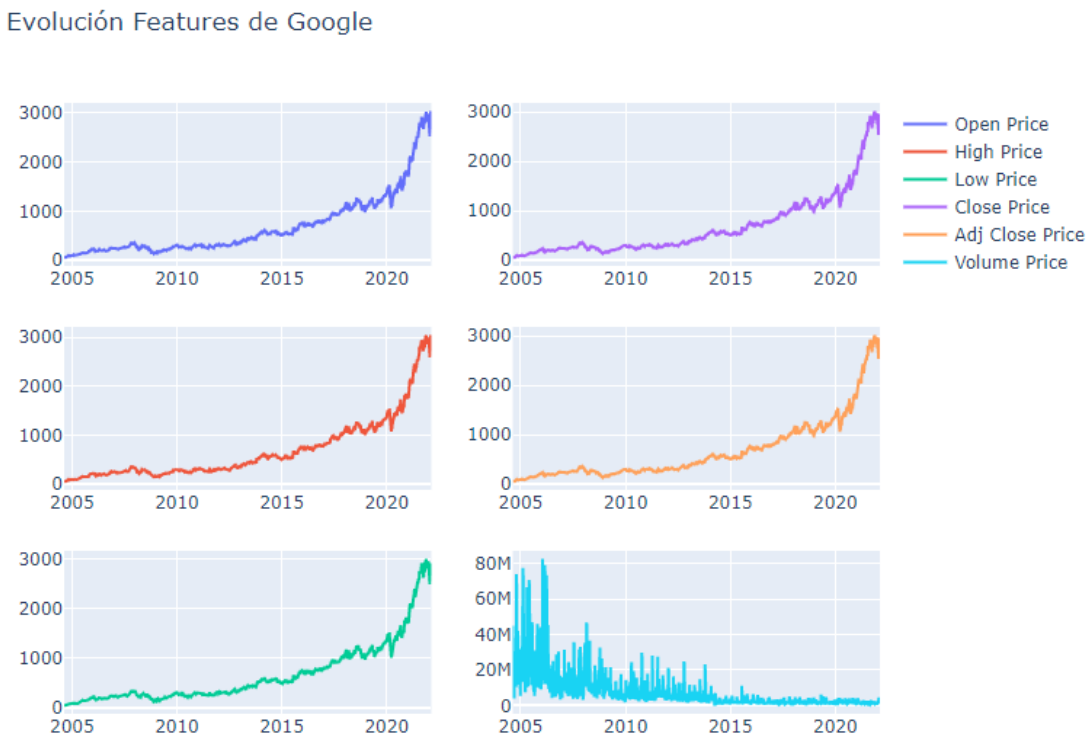
```
0
```

4.2. Exploratory Data Analysis

Ploteando cada una de las columnas:

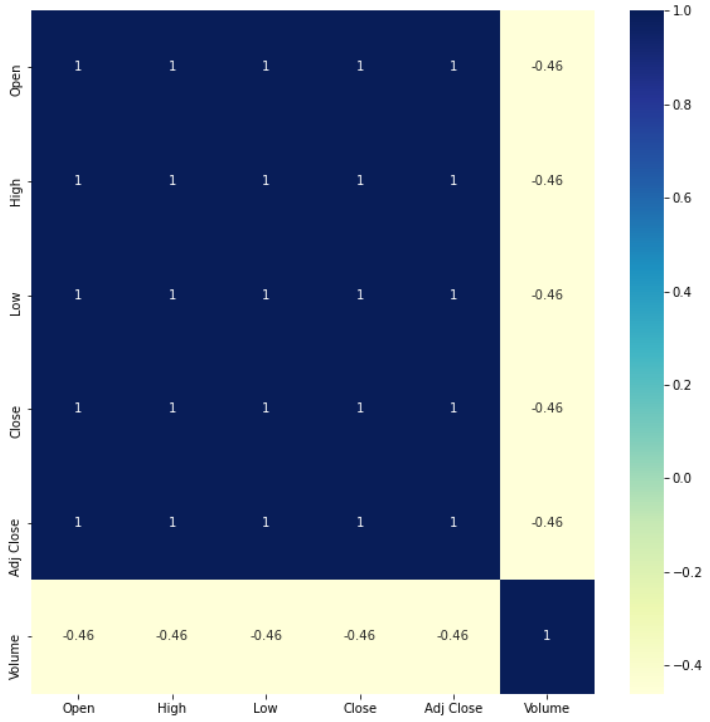


O con la librería **plotly**:



Como se observa a simple vista la evolución es indudablemente favorable en todas sus features.

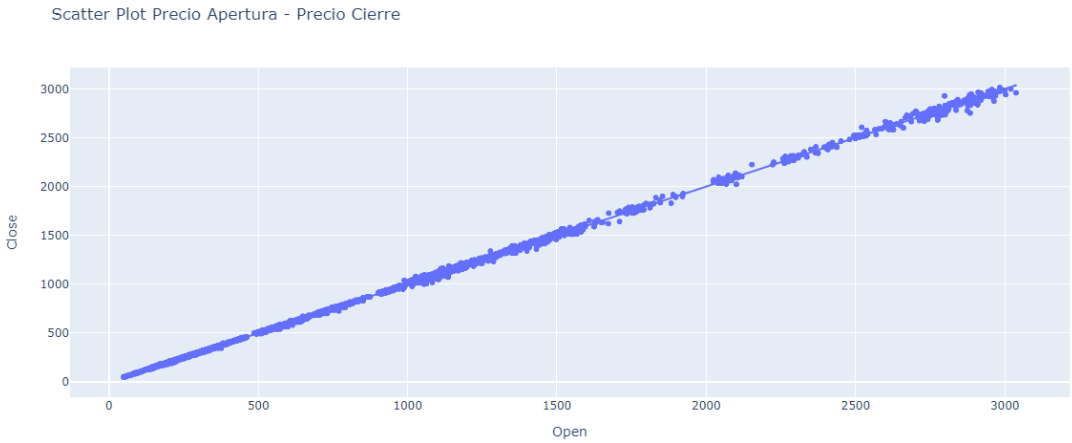
Además de la evolución temporal del precio, se va a estudiar la correlación entre las variables. Una primera aproximación puede ser un mapa de calor:



Como cabría esperar, todas las variables están fuertemente correlacionadas excepto la variable volumen que, aunque en parte también lo esté, su evolución no depende tanto del resto.

Ploteando precio de apertura vs precio de cierre (scatter plot):

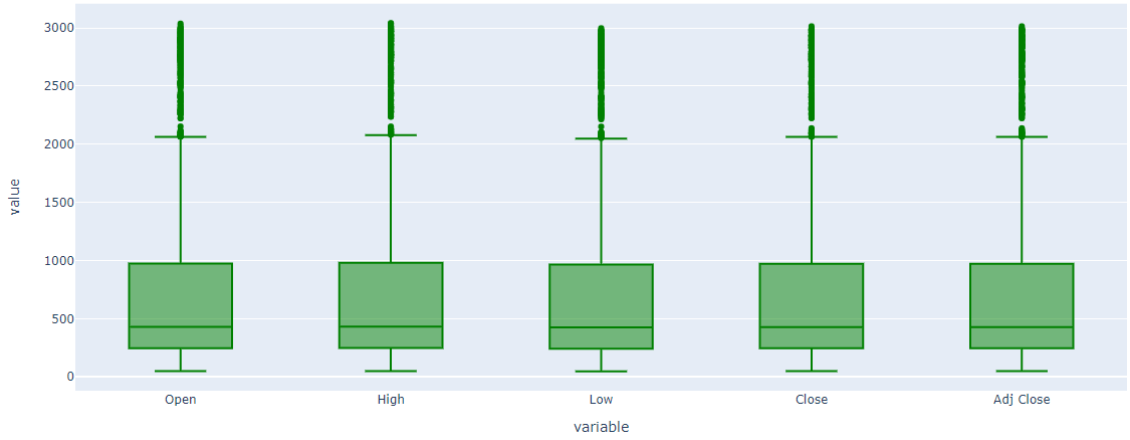
```
import plotly.express as px
fig = px.scatter(google, x="Open",
                 y="Close",
                 title = 'Scatter Plot Precio Apertura - Precio Cierre ',
                 trendline="ols")
fig.show()
```



Se obtiene una tendencia lineal ascendente con una pendiente de 0,5. Es decir, la linealidad de este gráfico es, prácticamente perfecta.

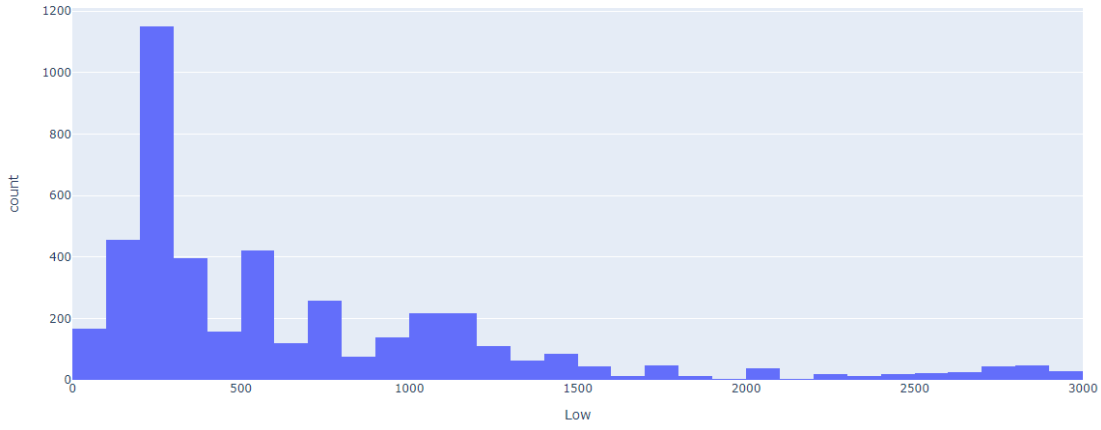
Otro método de estudio puede ser el diagrama de cajas y bigotes, el cual suele ser útil para comparar medias o desviaciones:

```
# Boxplots
google_box = google.drop(columns = 'Volume')
fig = px.box(google_box, color_discrete_sequence = ['green'])
fig.show()
```



Como las variables están fuertemente relacionadas este tipo de gráfico carece de sentido, ya que los resultados de cualquiera de las variables con respecto al precio es muy similar.

La representación de histogramas tampoco será muy útil debido a que los valores más antiguos tienen la misma influencia en el resultado que los nuevos, por lo que no arroja información útil a este caso práctico:



4.3. Análisis financiero

Haciendo una revisión algo más técnica se ahondará en temas económicos tales como dividendos, beneficios e inversores.

Se carga el activo en cuestión:

```
# Para ello debemos cargar de nuevo el activo
google_finance = yf.Ticker('GOOG')
google_finance
```

```
yfinance.Ticker object <GOOG>
```

```
#Institutional holders
holders_institutional = google_finance.institutional_holders

holders_institutional
```

	Holder	Shares	Date Reported	% Out	Value
0	Vanguard Group, Inc. (The)	21035884	2021-12-30	0.0666	60869223583
1	Blackrock Inc.	19196177	2021-12-30	0.0608	55545865805
2	Price (T.Rowe) Associates Inc	13325257	2021-12-30	0.0422	38557830402
3	State Street Corporation	10608366	2021-12-30	0.0336	30696261773
4	FMR, LLC	7678679	2021-12-30	0.0243	22218948767
5	Geode Capital Management, LLC	4669682	2021-12-30	0.0148	13512145138
6	Capital International Investors	4075940	2021-12-30	0.0129	11794099224
7	JP Morgan Chase & Company	3752343	2021-12-30	0.0119	10857742181
8	Northern Trust Corporation	3392530	2021-12-30	0.0107	9816590882
9	AllianceBernstein, L.P.	3082944	2021-12-30	0.0098	8920775928

Esta primera aproximación arroja información acerca de quién es el holder que más acciones tiene de Google (en Septiembre de 2021 Vanguard Group con un 6.6% del valor de la compañía) seguido del resto en orden descendente (suelen ser grandes compañías inversoras y no personas individuales como cabría esperar).

Con las recomendaciones de Google se puede consultar la opinión de grandes expertos en el sector para apoyarse y decidirse o no a invertir:

```
#Google Recomendaciones
recomendaciones = google_finance.recommendations

recomendaciones
```

Date	Firm	To Grade	From Grade	Action
2012-03-14 15:28:00	Oxen Group	Hold		init
2012-03-28 06:29:00	Citigroup	Buy		main
2012-04-03 08:45:00	Global Equities Research	Overweight		main
2012-04-05 06:34:00	Deutsche Bank	Buy		main
2012-04-09 06:03:00	Pivotal Research	Buy		main
...	...	...	...	...
2021-10-27 12:11:57	Oppenheimer	Outperform		main
2022-02-02 15:29:51	Raymond James	Outperform		main
2022-02-02 16:41:47	Credit Suisse	Outperform		main
2022-02-02 19:40:40	JP Morgan	Overweight		main
2022-02-02 19:43:53	Jefferies	Buy		main

Haciendo una agrupación del output anterior:

```
total_recomendaciones = google_finance .recommendations['To Grade'].value_counts()

total_recomendaciones
```

```
Buy                  105
Outperform            57
Overweight            41
Neutral               18
Hold                  11
Perform                4
Positive               3
Market Outperform      3
                       2
Long-Term Buy          1
Equal-weight           1
Market Perform         1
Strong Buy             1
Sell                   1
Name: To Grade, dtype: int64
```

La mayoría de los inversores recomienda comprar en ese determinado momento (siendo Outperform la sugerencia de compra moderada y Overweight la opinión de que dicha acción se revalorizará en un medio plazo).

Otra herramienta muy útil es “Google Noticias”. A través de esta interfaz se tiene acceso a las novedades más recientes de la compañía e incluso del mercado.

```
noticias = google_finance.news

noticias
```

```
[{'uuid': '80e7eb03-4fa2-3977-aabc-938e4ebb6599',
  'title': 'Frances Haugen: I believe in crypto',
  'publisher': 'Yahoo Finance Video',
  'link': 'https://finance.yahoo.com/video/frances-haugen-believe-crypto-110000416.html',
  'providerPublishTime': 1645095600,
  'type': 'VIDEO'},
 {'uuid': '8a73aef2-31dd-34b8-9c01-4eb4d4f8b69e',
  'title': 'Influencers with Andy Serwer: Frances Haugen',
  'publisher': 'Yahoo Finance Video',
  'link': 'https://finance.yahoo.com/video/influencers-andy-serwer-frances-haugen-110000589.html',
  'providerPublishTime': 1645095600,
  'type': 'VIDEO'},
 {'uuid': 'f5beedda-19e1-31e0-a552-1a5618d1aad0',
  'title': 'Why Frances Haugen wants ‘nutrition labels’ for social media platforms',
  'publisher': 'Yahoo Finance Video',
  'link': 'https://finance.yahoo.com/video/why-frances-haugen-wants-nutrition-110000522.html',
  'providerPublishTime': 1645095600,
```

Si se consulta, por ejemplo, las ganancias de Google en los últimos años:

```
ganancias = google_finance.earnings

ganancias
```

Year	Revenue	Earnings
2018	136819000000	30736000000
2019	161857000000	34343000000
2020	182527000000	40269000000
2021	257637000000	76033000000

Cuando se habla del término “Balance de Situación” se hace referencia al informe financiero contable que refleja la situación económica y financiera de una empresa en un determinado momento. Los activos son todos los bienes y derechos que una empresa posee y el pasivo hace referencia a las deudas y obligaciones de la empresa:

```
balance = google_finance.balance_sheet

balance
```

	2021-12-31	2020-12-31	2019-12-31	2018-12-31
Intangible Assets	1.417000e+09	1.445000e+09	1.979000e+09	2.220000e+09
Total Liab	1.076330e+11	9.707200e+10	7.446700e+10	5.516400e+10
Total Stockholder Equity	2.516350e+11	2.225440e+11	2.014420e+11	1.776280e+11
Other Current Liab	2.920800e+10	2.800600e+10	2.215900e+10	1.761200e+10
Total Assets	3.592680e+11	3.196160e+11	2.759090e+11	2.327920e+11
Common Stock	6.177400e+10	5.851000e+10	5.055200e+10	4.504900e+10
Other Current Assets	7.054000e+09	5.490000e+09	4.412000e+09	4.236000e+09
Retained Earnings	1.914840e+11	1.634010e+11	1.521220e+11	1.348850e+11
Other Liab	1.717300e+10	1.516000e+10	1.447800e+10	1.653200e+10
Good Will	2.295600e+10	2.117500e+10	2.062400e+10	1.788800e+10
Treasury Stock	-1.623000e+09	6.330000e+08	-1.232000e+09	-2.306000e+09
Other Assets	6.645000e+09	5.037000e+09	3.063000e+09	3.430000e+09
Cash	2.094500e+10	2.646500e+10	1.849800e+10	1.670100e+10
Total Current Liabilities	6.425400e+10	5.683400e+10	4.522100e+10	3.462000e+10
Deferred Long Term Asset Charges	1.284000e+09	1.084000e+09	7.210000e+08	7.370000e+08
Other Stockholder Equity	-1.623000e+09	6.330000e+08	-1.232000e+09	-2.306000e+09
Property Plant Equipment	1.105580e+11	9.696000e+10	8.458700e+10	5.971900e+10
Total Current Assets	1.881430e+11	1.742960e+11	1.525780e+11	1.356760e+11
Long Term Investments	2.954900e+10	2.070300e+10	1.307800e+10	1.385900e+10
Net Tangible Assets	2.272620e+11	1.999240e+11	1.788390e+11	1.575200e+11

La cuenta de resultados, en cambio, es un estado financiero que muestra ordenada y detalladamente la forma de cómo se obtuvo el resultado del ejercicio anterior. Como se observa en la siguiente tabla, el beneficio neto (Net Income) es creciente con el paso de los años:

```
cashflow = google_finance.cashflow

cashflow
```

	2021-12-31	2020-12-31	2019-12-31	2018-12-31
Investments	-8.806000e+09	-9.822000e+09	-4.017000e+09	-1.972000e+09
Change To Liabilities	1.057000e+09	1.329000e+09	4.650000e+08	1.438000e+09
Total Cashflows From Investing Activities	-3.552300e+10	-3.277300e+10	-2.949100e+10	-2.850400e+10
Net Borrowings	-1.236000e+09	9.661000e+09	-2.680000e+08	-6.100000e+07
Total Cash From Financing Activities	-6.136200e+10	-2.440800e+10	-2.320900e+10	-1.317900e+10
Change To Operating Activities	7.140000e+09	5.813000e+09	7.822000e+09	7.890000e+09
Net Income	7.603300e+10	4.026900e+10	3.434300e+10	3.073600e+10
Change In Cash	-5.520000e+09	7.967000e+09	1.797000e+09	5.986000e+09
Repurchase Of Stock	-5.027400e+10	-3.114900e+10	-1.839600e+10	-9.075000e+09
Effect Of Exchange Rate	-2.870000e+08	2.400000e+07	-2.300000e+07	-3.020000e+08
Total Cash From Operating Activities	9.165200e+10	6.512400e+10	5.452000e+10	4.797100e+10
Depreciation	1.243000e+10	1.367900e+10	1.165100e+10	9.029000e+09
Other Cashflows From Investing Activities	5.410000e+08	6.800000e+07	5.890000e+08	9.800000e+07
Change To Account Receivables	-9.095000e+09	-6.524000e+09	-4.340000e+09	-2.169000e+09
Other Cashflows From Financing Activities	-9.852000e+09	-2.920000e+09	-4.545000e+09	-4.043000e+09
Change To Netincome	4.712000e+09	9.349000e+09	7.707000e+09	3.298000e+09
Capital Expenditures	-2.464000e+10	-2.228100e+10	-2.354800e+10	-2.513900e+10

Otro dato de interés, por ejemplo, puede ser la inversión en I+D (Research Development). Cada ejercicio va siendo mejor que el anterior.

El flujo de caja es la acumulación neta de activos en un periodo determinado (indicador de gran importancia en la salud financiera de una empresa):

```
cashflow = google_finance.cashflow

cashflow
```

	2021-12-31	2020-12-31	2019-12-31	2018-12-31
Investments	-8.806000e+09	-9.822000e+09	-4.017000e+09	-1.972000e+09
Change To Liabilities	1.057000e+09	1.329000e+09	4.650000e+08	1.438000e+09
Total Cashflows From Investing Activities	-3.552300e+10	-3.277300e+10	-2.949100e+10	-2.850400e+10
Net Borrowings	-1.236000e+09	9.661000e+09	-2.680000e+08	-6.100000e+07
Total Cash From Financing Activities	-6.136200e+10	-2.440800e+10	-2.320900e+10	-1.317900e+10
Change To Operating Activities	7.140000e+09	5.813000e+09	7.822000e+09	7.890000e+09
Net Income	7.603300e+10	4.026900e+10	3.434300e+10	3.073600e+10
Change In Cash	-5.520000e+09	7.967000e+09	1.797000e+09	5.986000e+09
Repurchase Of Stock	-5.027400e+10	-3.114900e+10	-1.839600e+10	-9.075000e+09
Effect Of Exchange Rate	-2.870000e+08	2.400000e+07	-2.300000e+07	-3.020000e+08
Total Cash From Operating Activities	9.165200e+10	6.512400e+10	5.452000e+10	4.797100e+10
Depreciation	1.243000e+10	1.367900e+10	1.165100e+10	9.029000e+09
Other Cashflows From Investing Activities	5.410000e+08	6.800000e+07	5.890000e+08	9.800000e+07
Change To Account Receivables	-9.095000e+09	-6.524000e+09	-4.340000e+09	-2.169000e+09
Other Cashflows From Financing Activities	-9.852000e+09	-2.920000e+09	-4.545000e+09	-4.043000e+09
Change To Netincome	4.712000e+09	9.349000e+09	7.707000e+09	3.298000e+09
Capital Expenditures	-2.464000e+10	-2.228100e+10	-2.354800e+10	-2.513900e+10

Ahora se procede a la comparación de la evolución del precio de las acciones entre Google y otras empresas del mundo del BigTech (Microsoft, Apple,…):

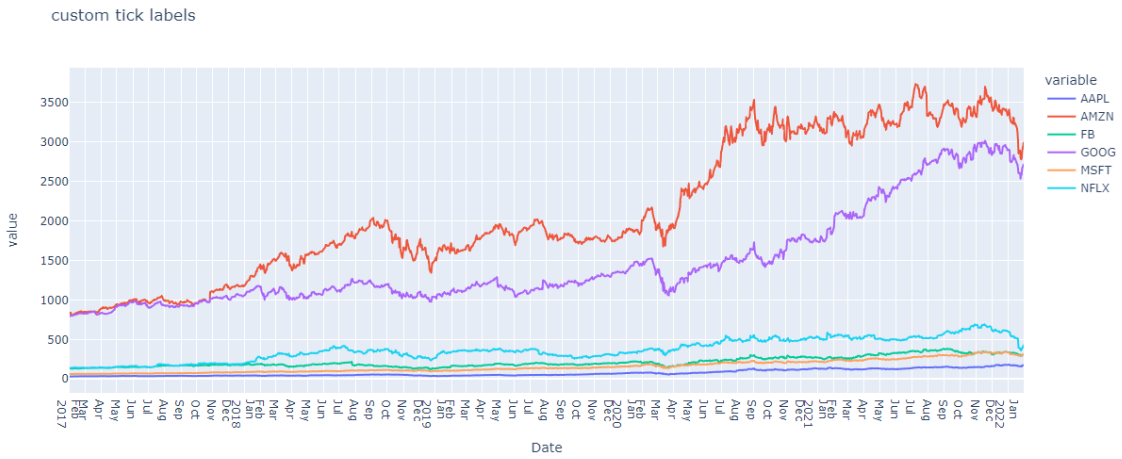
```
stocks = ['msft', 'aapl', 'nflx', 'goog', 'amzn', 'fb']

big_tech = yf.download(tickers=stocks, start='2017-02-01' ,end='2022-02-01', interval = '1d')
```

```
[**********100%**********]  6 of 6 completed
```

```
big_tech.head()
```

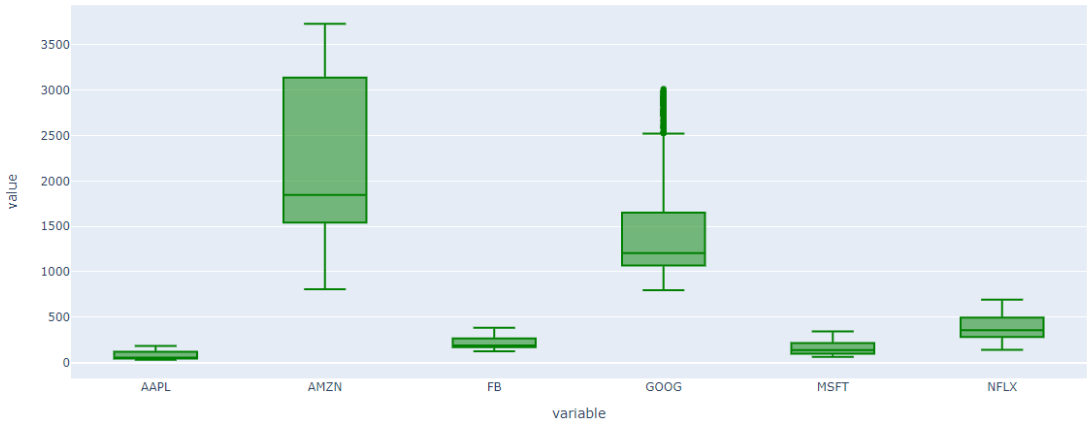
	Adj Close						Close				...	Open					
	AAPL	AMZN	FB	GOOG	MSFT	NFLX	AAPL	AMZN	FB	GOOG	...	FB	GOOG	MSFT	NFLX	AAPL	AMZN
Date																	
2017-01-31	28.519800	823.479980	130.320007	796.789978	60.110153	140.710007	30.337500	823.479980	130.320007	796.789978	...	130.169998	796.859985	64.860001	140.550003	196804000	3137200
2017-02-01	30.258953	832.349976	133.229996	795.695007	59.115299	140.779999	32.187500	832.349976	133.229996	795.695007	...	132.250000	799.679993	64.360001	141.199997	447940000	3850200
2017-02-02	30.207254	839.950012	130.839996	798.530029	58.734077	139.199997	32.132500	839.950012	130.839996	798.530029	...	133.220001	793.799988	63.250000	140.610001	134841600	7350500
2017-02-03	30.336515	810.200012	130.979996	801.489990	59.208275	140.250000	32.270000	810.200012	130.979996	801.489990	...	131.240005	802.989990	63.500000	139.509995	98029200	10868800
2017-02-06	30.620888	807.640015	132.059998	801.340027	59.171082	140.970001	32.572498	807.640015	132.059998	801.340027	...	130.979996	799.700012	63.500000	140.000000	107383600	3897300

Si se grafican todos los valores en una misma gráfica la cuantificación entre ellas será inmediata:



Haciendo una comparación entre la empresa más saludable hoy en día económicamente (Amazon) con Google se observa que su valor en 2017 era muy similar y, con el paso del tiempo, Amazon se ha revalorizado más y más rápido que Google.

El diagrama de cajas, en este caso, sí puede resultar útil:



NOTA: VER ARCHIVO HTML 001 EDA

5. FEATURE ENGINEERING

- En esta parte se realizará la ingeniería de las variables.
- En primer lugar se cargarán de nuevo los datos del histórico de las acciones de Google.
- Al tener 4394 filas y 6 columnas, se crearán nuevas variables con el objetivo de añadir más información que pueda ser relevante a la hora de modelar.
- Se harán transformaciones en las variables numéricas y categóricas (estas últimas serán añadidas en este paso).

Una vez cargados los datos:

```
# Primeras 5 filas del dataset
google.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2004-08-19	49.813290	51.835709	47.800831	49.982655	49.982655	44871361
1	2004-08-20	50.316402	54.336334	50.062355	53.952770	53.952770	22942874
2	2004-08-23	55.168217	56.528118	54.321388	54.495735	54.495735	18342897
3	2004-08-24	55.412300	55.591629	51.591621	52.239197	52.239197	15319808
4	2004-08-25	52.284027	53.798351	51.746044	52.802086	52.802086	9232276

Creamos 4 nuevas variables que se adjuntan al conjunto de datos.

Increase_Decrease: Esta variable calcula si hay un incremento del **volumen** respecto al día anterior. Si incrementa el **volumen** es 1, si disminuye el volumen es 0.

Buy_Sell_on_Open: Cuando el precio de **apertura** sea mayor al del día anterior el valor es 1, cuando el precio de **apertura** sea menor al del día anterior será 0.

Buy_Sell: Cuando el precio de **cierre ajustado** sea mayor al del día anterior el valor es 1, cuando el precio de **cierre ajustado** sea menor al del día anterior será 0.

Returns: Calcula el porcentaje de cambio en el precio con respecto al día anterior.

Veamos el código:

```
google['Increase_Decrease'] = np.where(google['Volume'].shift(-1) > google['Volume'],1,0) # Incremento del volumen, 1 si aumenta 0 si baja

google['Buy_Sell_on_Open'] = np.where(google['Open'].shift(-1) > google['Open'],1,0) # Si la apertura es mayor 1, si es menor 0

google['Buy_Sell'] = np.where(google['Adj Close'].shift(-1) > google['Adj Close'],1,0) # Si el cierre ajustado es mayor 1, si es menor 0

google['Returns'] = google['Adj Close'].pct_change() # Porcentaje de cambio
```

Primeras filas del nuevo dataset:

```
#Nuevo dataset de Google
google.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume	Increase_Decrease	Buy_Sell_on_Open	Buy_Sell	Returns
0	2004-08-19	49.813290	51.835709	47.800831	49.982655	49.982655	44871361	0	1	1	NaN
1	2004-08-20	50.316402	54.336334	50.062355	53.952770	53.952770	22942874	0	1	1	0.079430
2	2004-08-23	55.168217	56.528118	54.321388	54.495735	54.495735	18342897	0	1	0	0.010064
3	2004-08-24	55.412300	55.591629	51.591621	52.239197	52.239197	15319808	0	0	1	-0.041408
4	2004-08-25	52.284027	53.798351	51.746044	52.802086	52.802086	9232276	0	0	1	0.010775

Como se mencionó anteriormente, muchas de las variables, están correlacionadas, lo que haremos será quedarnos con aquellas con no lo estén tanto para asi evitar hacer trampas a la hora de modelar.

Código:

```
#Al haber muchas variables correladas nos quedamos solo algunas de ellas

google = google[['Date','Adj Close','Volume','Buy_Sell','Returns','Buy_Sell_on_Open','Increase_Decrease']]
```

Importante:

Cabe destacar que en cuanto a la construcción de modelos no centraremos sobretodo en aplicar **técnicas de forecasting** las cuales solo requeriran las columnas **Date** y **Adj Close** (Precio de cierre), sin embargo, probaremos con las variables creadas **modelos de regresión**, por lo que haremos para ellos una serie de **Pipelines** que servirán para hacer transformaciones en las variables numéricas y categóricas.

Dividimos los datos en Train,Test

```
X = google.drop('Adj Close',axis=1)

y = google['Adj Close']

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state = 1234)
```

5.1 CONSTRUCCIÓN DE PIPELINES

Los Pipeline resultan una herramienta crucial en el proceso de Machine Learning. Facilitan el proceso, por el hecho de que todos los pasos pueden agruparse de manera secuencial en otros pasos más genéricos.

Captura código:

```
# Transformador de variables numericas

transformador_numerico = Pipeline(steps=[

    # Paso 1: escalar las variables, nombre 'escalador'
    ('escalador', StandardScaler())])
```

```
# Transformador de variables categóricas: se encargará de imputar los valores perdidos de las variables categóricas y de

transformador_categorico = Pipeline(steps=[

    # Paso 1: codificación de variables categóricas, nombre 'onehot', usamos one hot encoding
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])
```

```
# Las numéricas
variables_numericas = google.select_dtypes(include=['int64', 'float64']).drop(['Adj Close'], axis=1).columns

# Las categóricas (excluida la dependiente)
variables_categoricas = google.select_dtypes(include=['object']).columns
```

```
#Construimos el procesador

preprocesador = ColumnTransformer(
    transformers=[
        # Primer transformador, lo llamamos numericas: aplica los pasos del trasnformador numerico
        # a las variables de 'variables_numericas'
        ('numericas', transformador_numerico, variables_numericas),

        # Segundo trasnformador, lo llamamos categoricas: aplica los pasos del transformacor categórico
        # a las variables de 'variables_categoricas'
        ('categoricas', transformador_categorico, variables_categoricas)])
```

NOTA VER ARCHIVO HTML : 002_FEATURENGINEERING

6.1 CONSTRUCCIÓN DE MODELOS.

En este apartado se procederá a la construcción de distintos modelos para hacer la predicción del precio de las acciones de los stocks.

La predicción de las series temporales puede clasificarse a grandes rasgos en las siguientes categorías:

Modelos clásicos/estadísticos - Medias móviles, alisamiento exponencial, ARIMA, SARIMA, TBATS

Machine learning- Regresión lineal, XGBoost, Random Forest, o cualquier modelo ML con métodos de reducción.

Deep Learning - RNN, LSTM,NeuralProphet.

6.1 Facebook Prophet

- La librería **Prophet** es una librería de código abierto diseñada para realizar forecasting de conjuntos de datos de series temporales univariantes. Es fácil de usar y está diseñada para encontrar automáticamente un buen conjunto de **hiper parámetros** para el modelo en un esfuerzo por hacer previsiones hábiles para datos con tendencias y estructura estacional por defecto.
- Esta basado en **Pytorch** paquete de Python empleado para **Deep Learning**.
- Es un modelo **fácil** de emplear pues no requiere una gran cantidad de código.

- Otra gran ventaja de este modelo es que los datos son **normalizados automáticamente** y en cuanto a la visualización, se puede graficar predicciones futuras y pasadas para ver una comparativa de cómo funciona el modelo.
- Así como también se pueden interpretar los **parámetros** y obtener su **descomposición** de manera gráfica.

Creación del modelo:

```
# ICreamos modelo y interval_width al 95%
my_model = Prophet(interval_width=0.95)
my_model.fit(data)
```

```
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
<fbprophet.forecaster.Prophet at 0x7fd28a840410>
```

```
[ ] # Creamos una variable con el número de predicciones y la frecuencia(en este caso diaria)
future_dates = my_model.make_future_dataframe(periods=365, freq='D')
future_dates.head()
```

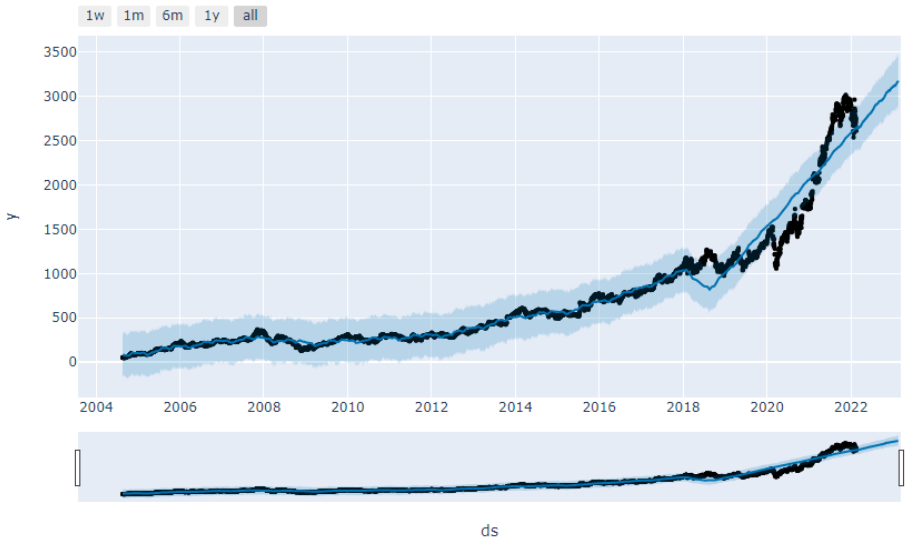
	ds
0	2004-08-19
1	2004-08-20
2	2004-08-23
3	2004-08-24
4	2004-08-25

```
[ ] # Forecssting del modelo
forecast = my_model.predict(future_dates)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

	ds	yhat	yhat_lower	yhat_upper
4768	2023-02-14	3166.089543	2887.780556	3442.610098
4769	2023-02-15	3168.421459	2860.139249	3446.629529
4770	2023-02-16	3169.672464	2891.526251	3458.520490
4771	2023-02-17	3170.609082	2886.456512	3478.521290

Gráfico Forecast

```
# Gráfico Forecast de manera interactiva con Plotly
plot2 = plot_plotly(my_model, forecast)
plot2
```



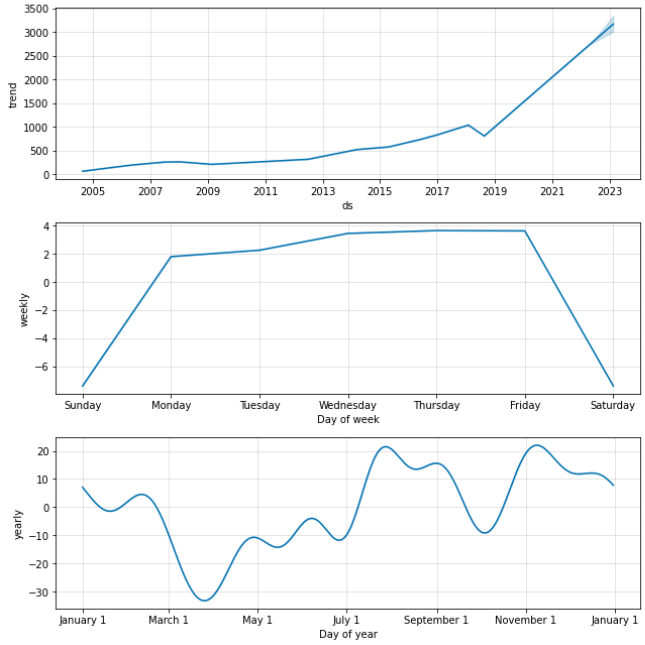
Prophet representa los valores observados de nuestras series temporales (**los puntos negros**), los valores previstos (**línea azul**) y los intervalos de incertidumbre de nuestras previsiones (**las regiones sombreadas en azul**).

Otra característica especialmente destacada de Prophet es su capacidad para devolver los **componentes** de nuestras previsiones.

Esto puede ayudar a revelar cómo los patrones diarios, semanales y anuales de las series temporales contribuyen a los valores generales previstos.

Gráfico de los componentes

```
fig1 = my_model.plot_components(forecast)
```



Evaluación del modelo

Para ver el MAE y RMSE, debemos divir los datos en train y test y hacer una validación cruzada.

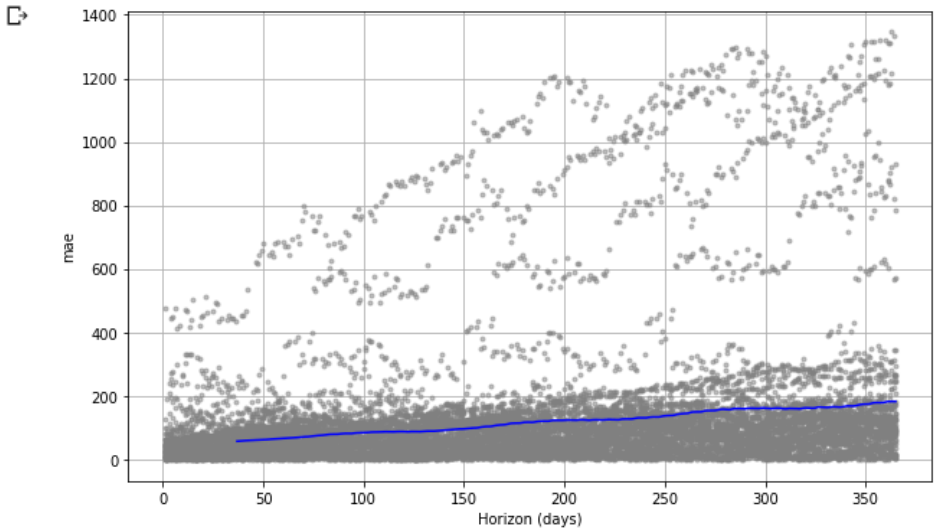
- Tenemos 4408 observaciones que van desde el 19 de agosto de 2004 hasta el 18 de febrero de 2022.
- Lo que usaremos será 80 % para train y 20 % test El 80% del train es hasta la observaciones 3526
- La función **cross_validation** realiza la separación de datos automaticamente, por lo que solo tenemos que llamarla.

```
[ ] from fbprophet.diagnostics import performance_metrics
    df_p = performance_metrics(df_cv)
    df_p.head()
```

	horizon	mse	rmse	mae	mape	mdape	coverage
0	37 days	9062.105791	95.195093	57.679475	0.100760	0.070128	0.537066
1	38 days	9110.099688	95.446842	58.231820	0.102235	0.071010	0.532670
2	39 days	9351.583081	96.703584	59.131118	0.103858	0.071659	0.524075
3	40 days	9401.473788	96.961197	59.387780	0.104762	0.071560	0.520858
4	41 days	9370.551418	96.801609	59.520915	0.105817	0.071560	0.518255

Realizamos un gráfico para ver la evolución del Error Medio (MAE)

```
from fbprophet.plot import plot_cross_validation_metric
fig = plot_cross_validation_metric(df_cv, metric='mae')
```



En este último gráfico podemos ver, cómo al hacer la validación cruzada con los diferentes horizontes, el error cuadrático. con el paso del tiempo tiende a ser mayor.

NOTA VER ARCHIVO HTML : 003_FBProphet

6.2 Time series con PYCARET

A finales del 2021, se hizo oficial la puesta en funcionamiento de una nueva librería para series temporales, que permite crear una gran cantidad de modelos de forecasting en pocas líneas de código.

Esta funcionalidad la hemos visto muy útil, puesto que permite generar modelos de forecast de una manera muy ágil y simple.

Beneficios de esta librería:

- Optimización de tiempo, el poder crear tantos modelos y hacer una comparación rápida permite tener una visión global de los que mejores dan resultados ganándose productividad y tiempo.

La función **set_up** incluye como parametros:

- **Data**: el dataset al que heremos realizarle el forecast.
- **Fh** : Es el horizonte o la cantidad de días en este caso que queremos predecir.
- **Fold**: consiste en tomar los datos originales y crear a partir de ellos dos conjuntos separados: un primer conjunto de entrenamiento (y prueba), y un segundo conjunto de validación.

Luego, el conjunto de entrenamiento se va a dividir en k subconjuntos y, al momento de realizar el entrenamiento, se va a tomar cada k subconjunto como conjunto de prueba del modelo, mientras que el resto de los datos se tomará como conjunto de entrenamiento.

Este proceso se repetirá k veces, y en cada iteración se seleccionará un conjunto de prueba diferente, mientras los datos restantes se emplearán, como se mencionó, como conjunto de entrenamiento

Con la función compare_models, nos da como salida todo tipo de modelos clásicos junto con sus métricas.

Ejemplo:

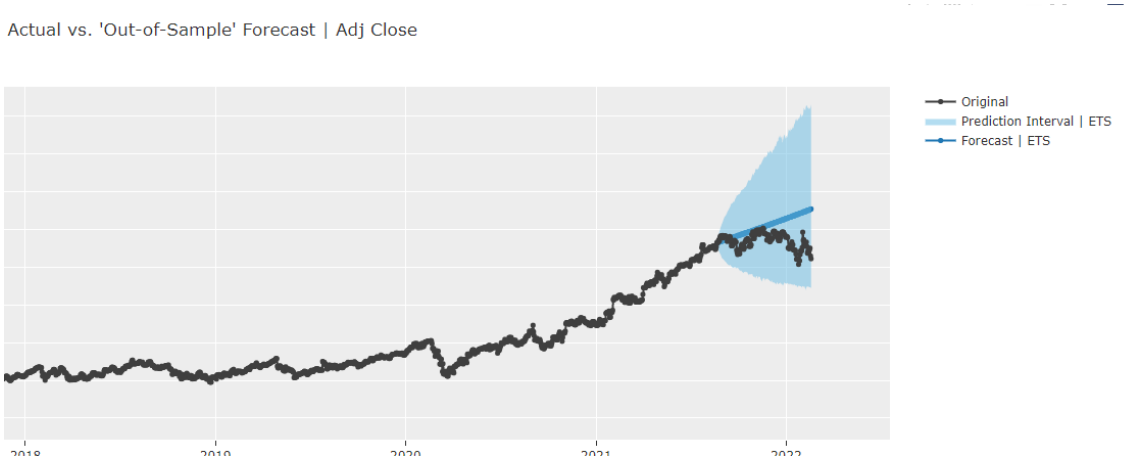
	Model	MAE	RMSE	MAPE	SMAPE	MASE	RMSSE	R2
exp_smooth	Exponential Smoothing	17.1926	20.1633	0.0435	0.0439	0.5852	0.6105	0.8918
ets	ETS	17.4165	20.5103	0.0440	0.0445	0.5931	0.6212	0.8882
et_cds_dt	Extra Trees w/ Cond. Deseasonalize & Detrending	19.6620	24.0121	0.0490	0.0489	0.6666	0.7255	0.8465
huber_cds_dt	Huber w/ Cond. Deseasonalize & Detrending	20.0334	25.9670	0.0491	0.0499	0.6813	0.7866	0.8113
arima	ARIMA	20.0069	22.2199	0.0501	0.0507	0.6830	0.6735	0.8677
lr_cds_dt	Linear w/ Cond. Deseasonalize & Detrending	20.6084	25.4401	0.0509	0.0514	0.7004	0.7702	0.8215
ridge_cds_dt	Ridge w/ Cond. Deseasonalize & Detrending	20.6086	25.4405	0.0509	0.0514	0.7004	0.7703	0.8215
lar_cds_dt	Least Angular Regressor w/ Cond. Deseasonalize & Detrending	20.6084	25.4401	0.0509	0.0514	0.7004	0.7702	0.8215
en_cds_dt	Elastic Net w/ Cond. Deseasonalize & Detrending	20.6816	25.5362	0.0511	0.0516	0.7029	0.7732	0.8201
lasso_cds_dt	Lasso w/ Cond. Deseasonalize & Detrending	20.7373	25.6005	0.0512	0.0517	0.7048	0.7751	0.8193
br_cds_dt	Bayesian Ridge w/ Cond. Deseasonalize & Detrending	20.9213	25.8795	0.0515	0.0521	0.7112	0.7837	0.8144

En nuestro caso, el modelo ganador ha sido un ETS, este modelo puede ser tuneado para intentar mejorar su acierto y después evaluar sus componentes.

```
[ ] # Model Tuning
    tuned_et = exp.tune_model(best)
```

	cutoff	MAE	RMSE	MAPE	SMAPE	MASE	RMSSE	R2
0	2019-09-02	102.6461	121.5058	0.0739	0.0777	4.6308	3.9786	-0.5315
1	2020-02-29	101.2858	139.2765	0.0821	0.0757	4.4551	4.4218	0.0666
2	2020-08-27	102.5623	133.9012	0.0599	0.0593	4.1474	3.8315	0.4695
3	2021-02-23	212.5867	249.4371	0.0838	0.0886	7.8039	6.2936	-0.1086
Mean	NaT	129.7702	161.0302	0.0749	0.0753	5.2593	4.6314	-0.0260
SD	NaT	47.8172	51.4469	0.0094	0.0105	1.4793	0.9840	0.3593

Gráfico Forecast con Intervalos



En el gráfico de arriba podemos ver el forecasting de nuestro mejor modelo con los intervalos de confianza del 95 %, vemos como respeta estos intervalos, sin embargo, nuestro modelo estima que el precio iba a ser mayor que el actual.

Predicciones:

```
exp.predict_model(tuned_et,return_pred_int=True)
```

	Model	MAE	RMSE	MAPE	SMAPE	MASE	RMSSE	R2
0	ETS	209.8865	274.0623	0.0758	0.0712	7.2788	6.5144	-6.1123

	y_pred	lower	upper
2021-08-23	2821.4875	2742.4137	2894.6398
2021-08-24	2825.0736	2719.9568	2929.0613
2021-08-25	2831.4512	2700.5505	2958.2224
2021-08-26	2834.0906	2685.2411	2982.8556
2021-08-27	2837.7895	2677.9599	3008.8192
...	...	...	...
2022-02-14	3249.5952	2242.4875	4583.7896
2022-02-15	3253.7253	2238.8734	4608.6426
2022-02-16	3261.0706	2230.6842	4588.0512
2022-02-17	3264.1105	2222.7213	4613.7444
2022-02-18	3268.3706	2228.7147	4650.7082

180 rows × 3 columns

NOTA: VER NOTEBOOK 004_TimeSeries_Pycaret

6.3 Neural Prophet

- El forecasting utilizando la red neuronal Prophet es un modelo híbrido (Lineal - Red Neuronal).
- Está basado en Pytorch paquete de Python empleado para Deep Learning.
- Es un modelo fácil de emplear pues no requiere una gran cantidad de código.
- Otra gran ventaja de este modelo es que los datos son normalizados automáticamente y en cuanto a la visualización, se puede graficar predicciones futuras y pasadas para ver una comparativa de cómo funciona el modelo.
- Asi como también interpretar los parámetros y obtener su descomposición de manera gráfica.

Ventajas de NeuralPropeht:

- NeuralProphet utiliza el descenso de gradiente de PyTorch para la optimización, lo que hace que el modelado sea mucho más rápido
- La autocorrelación de series de tiempo se modela utilizando la Red Auto-Regresiva.

- Los regresores rezagados se modelan utilizando una red neuronal Feed-Forward separada.

Entrenamos el modelo :

```
# Creo el modelo Neural Prophet y entreno el modelo
m = NeuralProphet()
model = m.fit(data, freq='D', epochs=1000)
```

```
INFO - (NP.utils.set_auto_seasonalities) - Disabling daily seasonality. Run NeuralProphet with daily_seasonality=True to override this.
INFO - (NP.config.set_auto_batch_epoch) - Auto-set batch_size to 32
98% 275/282 [00:03<00:00, 135.45it/s]
INFO - (NP.utils_torch.lr_range_test) - lr-range-test results: steep: 6.75E-02, min: 2.01E+00
99% 278/282 [00:01<00:00, 154.12it/s]
INFO - (NP.utils_torch.lr_range_test) - lr-range-test results: steep: 6.75E-02, min: 2.70E+00
99% 278/282 [00:01<00:00, 158.73it/s]
INFO - (NP.utils_torch.lr_range_test) - lr-range-test results: steep: 6.75E-02, min: 1.61E+00
INFO - (NP.forecaster._init_train_loader) - lr-range-test selected learning rate: 6.75E-02
Epoch[1000/1000]: 100%| 1000/1000 [04:26<00:00,  3.75it/s, SmoothL1Loss=0.00146, MAE=71.8, RMSE=116, RegLoss=0]
```

El **MAE** (Error medio del modelo) es el 71.8 y **RMSE** (Error cuadrático medio) es de 116

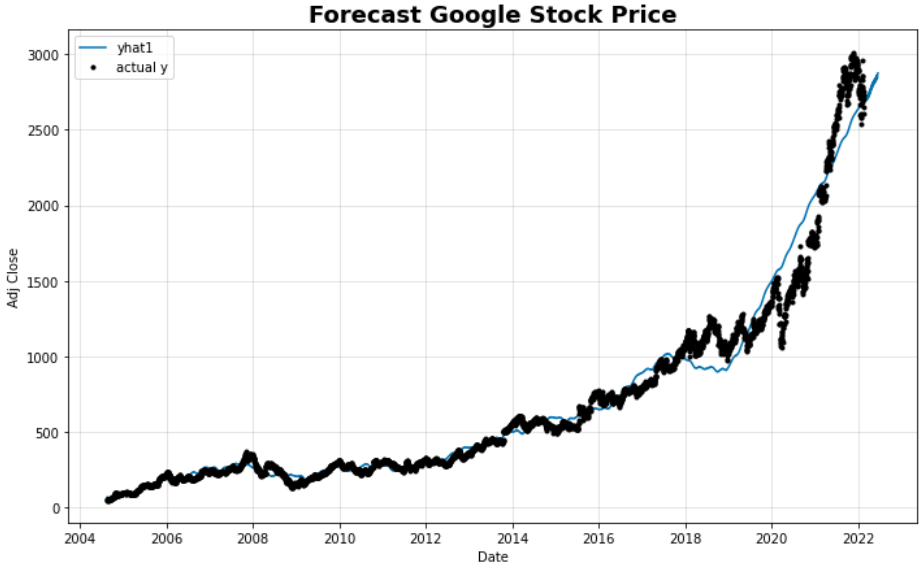
Es decir si nuestra predicción da un precio de 1500, nuestro modelo estará de media 71.8$ por encima o por debajo de ese precio

Se realizan predicciones de futuro y se grafica

```
[ ] # Se realizan las Predicciones del modelo
    future = m.make_future_dataframe(data, periods=120, n_historic_predictions=len(data))
    forecast = m.predict(future)
```

- En la tabla de arriba podemos ver la fecha , la y que es el valor del precio de cierre diario.
- yhat1 que corresponde a la predicción que ha hecho nuestro modelo.
- Así mismo,los valores de la tendencia, estacionalidad anual y semanal.

```
Text(0.5, 1.0, 'Forecast Google Stock Price')
```

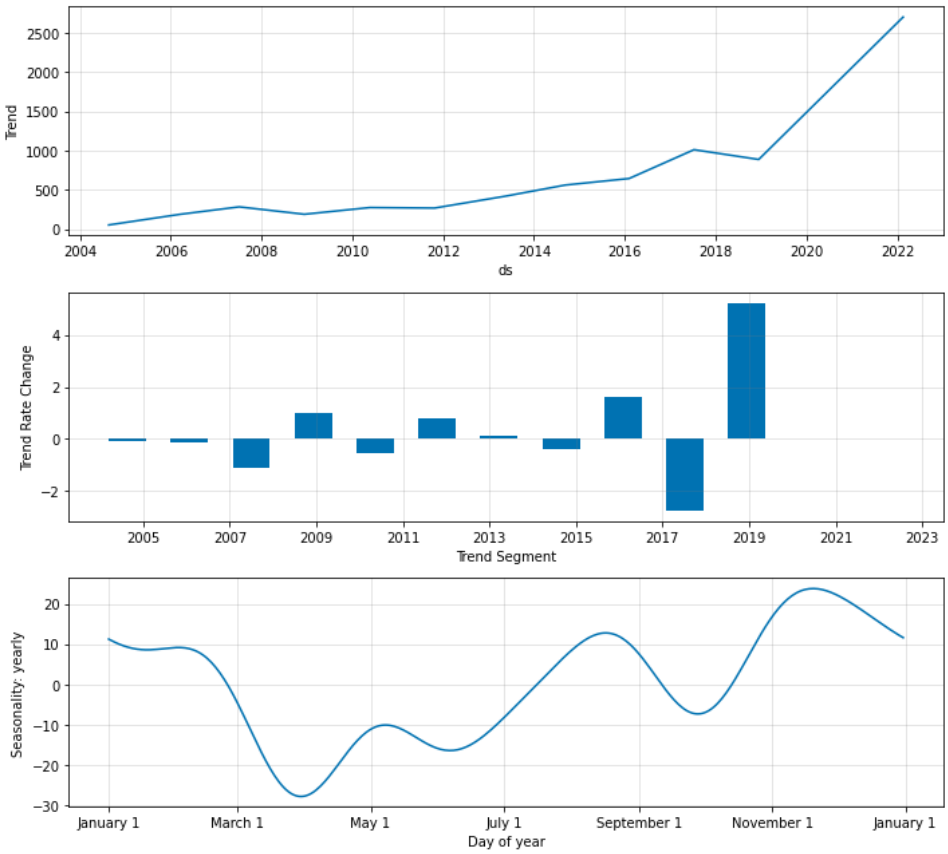


En el gráfico se puede ver la evolución que ha tenido el precio comparandose el valor real(y) y el predictivo(yhat1).

La parte la linea azul corresponde al valor predictivo del modelo y los puntos negros al valor real.

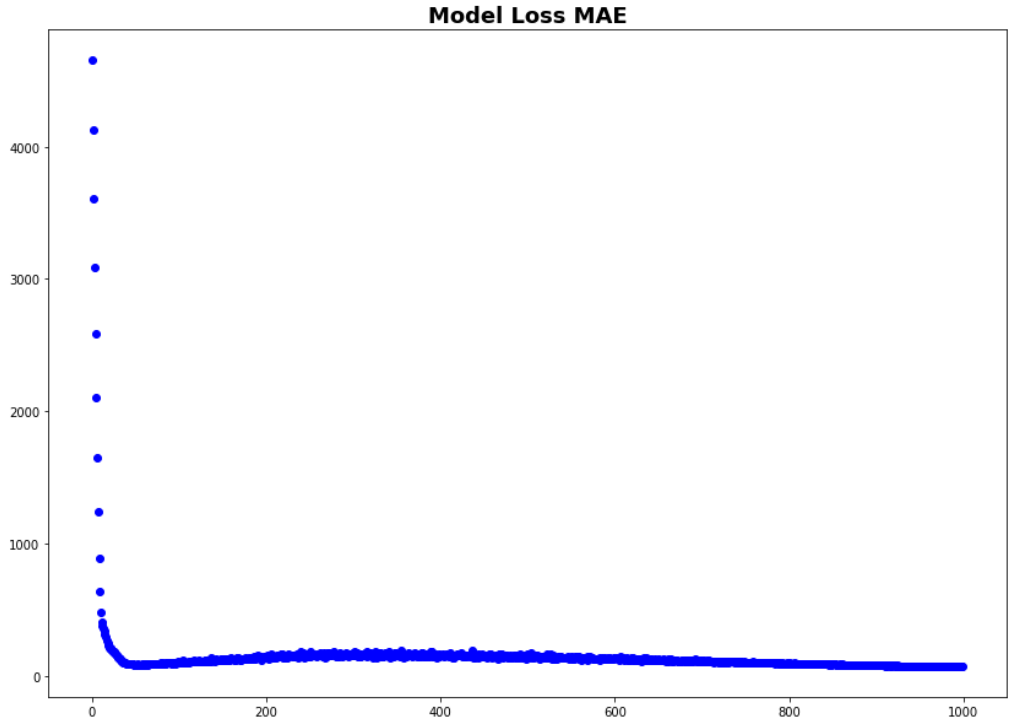
Asi mismo, se ha hecho una predicción de la temperatura desde 2016 hasta mediados del 2018.

Con la función **plot_components** vemos los diferentes componentes de la serie temporal, los residuos, la tendencia, la estacionalidad semanal y anual.



Para ver la evolución del error medio del modelo en los diferentes epochs hemos ploteado una gráfica que muestra dicha evolución

```
Text(0.5, 1.0, 'Model Loss MAE')
```



Así vemos, como en las primeras epochs el error es grande pero a medida que va realizando más interacciones el error se reduce hasta un 71.8 de MAE.

Información del modelo:

```
# Información de los errores del modelo
model
```

	SmoothL1Loss	MAE	RMSE	RegLoss
0	1.655361	4656.927609	5448.834376	0.0
1	1.421440	4123.873471	4882.618071	0.0
2	1.194611	3603.086326	4311.061833	0.0
3	0.975305	3090.825781	3744.233714	0.0
4	0.765192	2591.175171	3183.352894	0.0
...	...	...	...	...
995	0.001464	71.781709	115.884934	0.0
996	0.001464	71.782884	115.946848	0.0
997	0.001464	71.783281	116.344291	0.0
998	0.001464	71.780772	115.410033	0.0
999	0.001464	71.779883	116.058469	0.0

1000 rows × 4 columns

NOTA VER ARCHIVO HTML : 005_NeuralProphet

6.4 Regression Models with Pycaret

Aparte de la nueva librería de time series de pycaret, una de las más clásicas y más utilizadas ha sido la de regression puesto que esta contiene los modelos de machine learning más utilizados como pueden ser Lasso, Ridge, Random Forest, XGboost.

Para este caso, utilizaremos las siguientes variables:

	Date	Adj Close	Volume	Buy_Sell	Returns	Buy_Sell_on_Open	Increase_Decrease
0	2004-08-19	49.982655	44871361	1	NaN	1	0
1	2004-08-20	53.952770	22942874	1	0.079430	1	0
2	2004-08-23	54.495735	18342897	0	0.010064	1	0
3	2004-08-24	52.239197	15319808	1	-0.041408	0	0
4	2004-08-25	52.802086	9232276	1	0.010775	0	0

Hemos borrado algunas que estaban correlacionadas y en esta ocasión, sí aprovecharemos las variables creadas en el feature engineering.

Dividimos los datos en train y test:

```
data = google.sample(frac=0.8, random_state=1234).reset_index(drop=True)
data_unseen = google.drop(data.index).reset_index(drop=True)

print('Data for Modeling: ' + str(data.shape))
print('Unseen Data For Predictions: ' + str(data_unseen.shape))
```

```
Data for Modeling: (3527, 7)
Unseen Data For Predictions: (882, 7)
```

Iniciamos el set up poniendo como target 'Adj Close' y a las demás features les hacemos algunas transformaciones, también eliminamos la multicolinealidad (es la relación de dependencia lineal fuerte entre más de dos variables explicativas en una regresión múltiple).

```
xp_reg102 = setup(data = google, target = 'Adj Close', session_id=123,
                  normalize = True, transformation = True, transform_target = True,
                  combine_rare_levels = True, rare_level_threshold = 0.05,
                  remove_multicollinearity = True)
```

Una vez iniciado el setup, activamos la función compare_models

Esta función nos devolverá practicamente todos los modelos que hay de machine learning con sus metricas para poder comparar la efectividad entre ellos.

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	T
gbr	Gradient Boosting Regressor	231.7565	169063.8941	409.9158	0.5846	0.3990	0.3275	
lightgbm	Light Gradient Boosting Machine	238.6466	174888.8370	416.6831	0.5709	0.4109	0.3403	
rf	Random Forest Regressor	240.3001	176694.2122	419.0110	0.5659	0.4139	0.3406	
et	Extra Trees Regressor	247.4800	185675.2281	430.0889	0.5421	0.4299	0.3521	
ada	AdaBoost Regressor	258.6841	189957.4832	434.7813	0.5329	0.4680	0.3916	
omp	Orthogonal Matching Pursuit	254.4660	206648.3366	453.3066	0.4923	0.4481	0.3718	
lr	Linear Regression	255.2466	207664.7688	454.7445	0.4891	0.4442	0.3680	
lar	Least Angle Regression	255.2465	207664.7392	454.7444	0.4891	0.4442	0.3680	
ridge	Ridge Regression	255.2270	207705.5234	454.7870	0.4890	0.4442	0.3679	
br	Bayesian Ridge	255.1361	207921.2352	455.0123	0.4885	0.4442	0.3677	
huber	Huber Regressor	255.7042	217891.6124	465.7418	0.4642	0.4469	0.3591	
knn	K Neighbors Regressor	278.6361	242534.8346	491.4447	0.4033	0.4960	0.4150	
dt	Decision Tree Regressor	323.9558	326560.8652	570.0622	0.1934	0.5628	0.4787	
par	Passive Aggressive Regressor	333.9401	325372.1513	568.2699	0.1859	0.6550	0.4708	
en	Elastic Net	370.6830	381093.8031	616.0068	0.0626	0.7052	0.6295	
lasso	Lasso Regression	445.7993	458629.5562	676.0446	-0.1295	0.8689	0.8670	
llar	Lasso Least Angle Regression	445.7993	458629.5791	676.0446	-0.1295	0.8689	0.8670	
dummy	Dummy Regressor	445.7993	458629.5789	676.0446	-0.1295	0.8689	0.8670	

En este caso el modelo ganador ha sido el GradientBoosting, un algoritmo que en muchas ocasiones suele dar muy buenos resultados y suele ser un algoritmo top en la mayoria de concursos de machine learning.

Tuneamos el modelo con 50 interacciones.

```
#Tuneado con 50 interacciones
best_model_tuned = tune_model(best_model, n_iter = 50)
```

	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	230.0146	176688.5162	420.3433	0.5634	0.3945	0.3154
1	242.2149	166443.8793	407.9753	0.5763	0.3983	0.3428
2	236.4699	163672.5872	404.5647	0.5805	0.4307	0.3650
3	245.5740	195033.3007	441.6257	0.5189	0.4208	0.3330
4	211.1456	145246.4087	381.1121	0.5790	0.3836	0.3165
5	202.0847	126151.3722	355.1779	0.6495	0.3833	0.3277
6	250.2915	202221.0123	449.6899	0.5402	0.4130	0.3469
7	276.0177	227239.4122	476.6964	0.5445	0.4356	0.3449
8	226.7652	167421.8170	409.1721	0.5948	0.4042	0.3202
9	255.0851	185757.7914	430.9963	0.5401	0.4211	0.3500
Mean	237.5663	175587.6097	417.7354	0.5687	0.4085	0.3362
SD	20.4083	27416.8053	32.9358	0.0350	0.0177	0.0155

Tras el tuneado y aplicando la función finalize_model que termina de hacer ajustes en él para mejorarlo aún más, logramos un R2 del 0.71 en train y test.

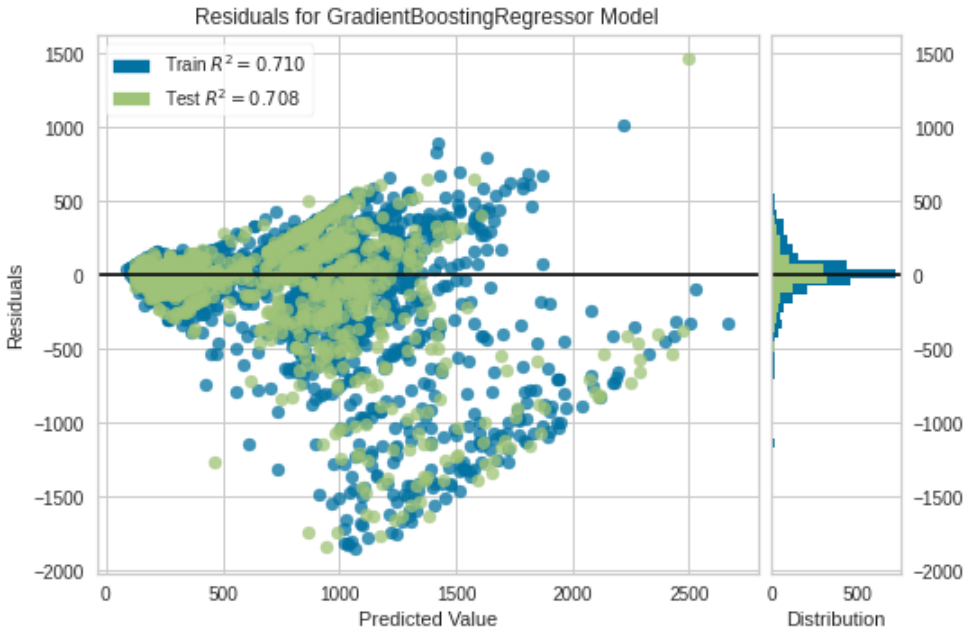
R-cuadrado = Variación explicada / variación total

El R-cuadrado siempre está entre 0 y 100%:

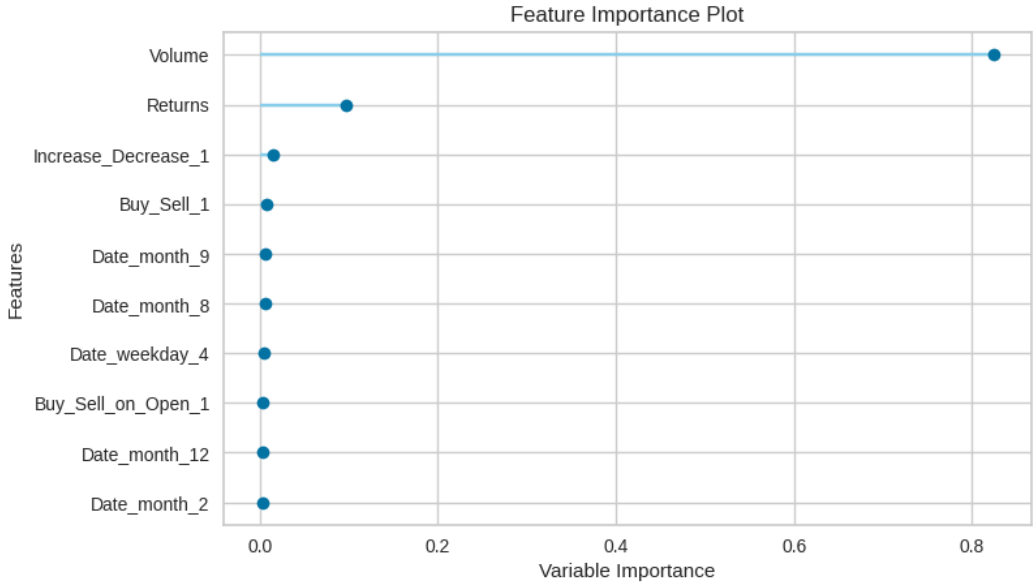
0% indica que el modelo no explica ninguna porción de la variabilidad de los datos de respuesta en torno a su media.

100% indica que el modelo explica toda la variabilidad de los datos de respuesta en torno a su media. En general, cuanto mayor es el R-cuadrado, mejor se ajusta el modelo a los datos. Sin embargo, hay condiciones importantes con respecto a esta pauta de las que hablaré más adelante.

```
#Plot model
plot_model(final_gbm)
```



Por último, creamos un gráfico para ver la importancia de las variables.



NOTA VER ARCHIVO HTML : 006_RegressionModels

6.5 Modelo LSTM

6.6 Comparativa Resultados Modelos

7. API DEPLOYMENT

El ciclo de vida de un proyecto de machine learning tiene como uno de los últimos pasos el despliegue del modelo o también conocido como la parte del “Deployment”.

Esta fase es de suma importancia y muchas empresas están teniendo problemas en ella, puesto que una vez realizado todos los modelos de machine learning, es importante productivizar esos modelos y que generen un valor económico a la empresa (aumento de beneficios, reducción de costes, anticipación de escenarios).

Puedes tener un modelo con muy buenas métricas pero si la empresa no es capaz de llevarlo a producción de poco habrá servido todas las demás fases. Por otro lado, es importante que sea una aplicación o producto que pueda ser utilizado por cualquier usuario de una manera intuitiva y sencilla, permitiendo así una máxima utilidad y accesibilidad para las aréas de negocio.

En nuestro proyecto, al tratarse de predicción de precios de stocks financieros. Hemos decidido crear una API a través de **Streamlit.**

Streamlit es una librería que, de forma sencilla, te permite crear todo tipo de aplicaciones de datos desarrolladas en Python. Es gratis y además muy intuitiva. Con ella, se pueden crear API que podrán ser compartidas con aréas de negocio, clientes y usuarios finales.

Con el servicio Streamlit Sharing, se permite subir a través de una cuenta GitHub la aplicación a la nube de Streamlit, así que ya no solo estaría la API accesible a través de una máquina local, sino que al subirla a la nube, cualquier usuario podrá acceder a ella con el enlace web y utilizarla sin ningún coste ni tener que instalar nada.

La API ha sido desarrollada en el edito de código fuente Visual Studio Code, concretamente en un script en lenguaje Python.

https://share.streamlit.io/sebasespon/api-forecast-stocks/main/main.py

Para crear la API, hemos utilizado la libreria Streamlit, Plotly para las visualizaciones, Pandas para algunas estadísticas, Yfinance para cargar los datos de los activos financieros y Prophet la libreria de forecast de Facebook.

En la carpeta API (adjuntada en los archivos del proyecto) podrán encontrar:

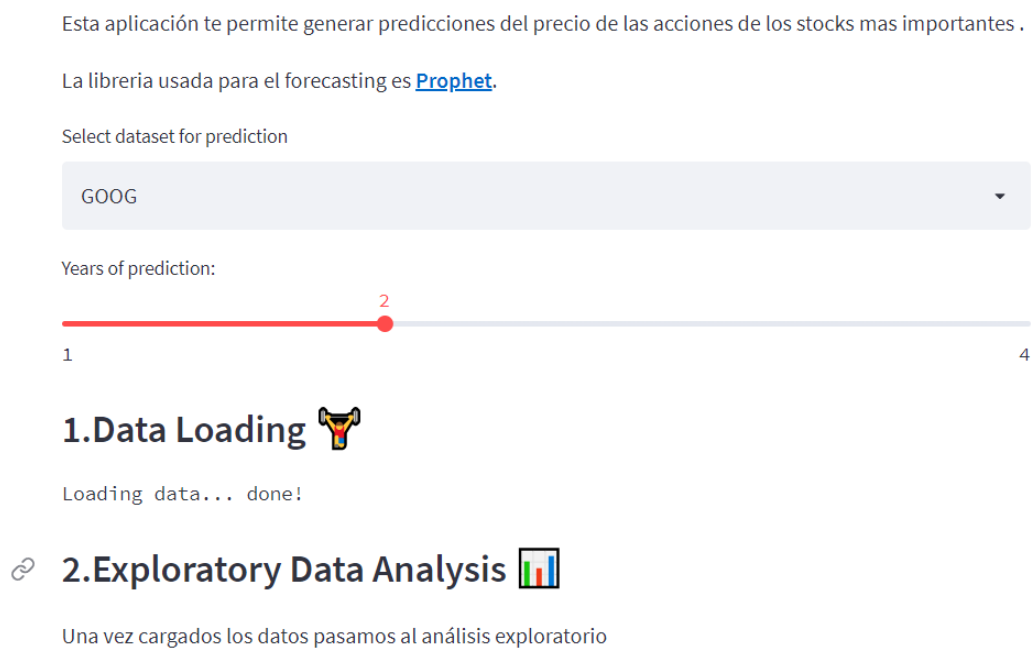
El script en Python : main.py

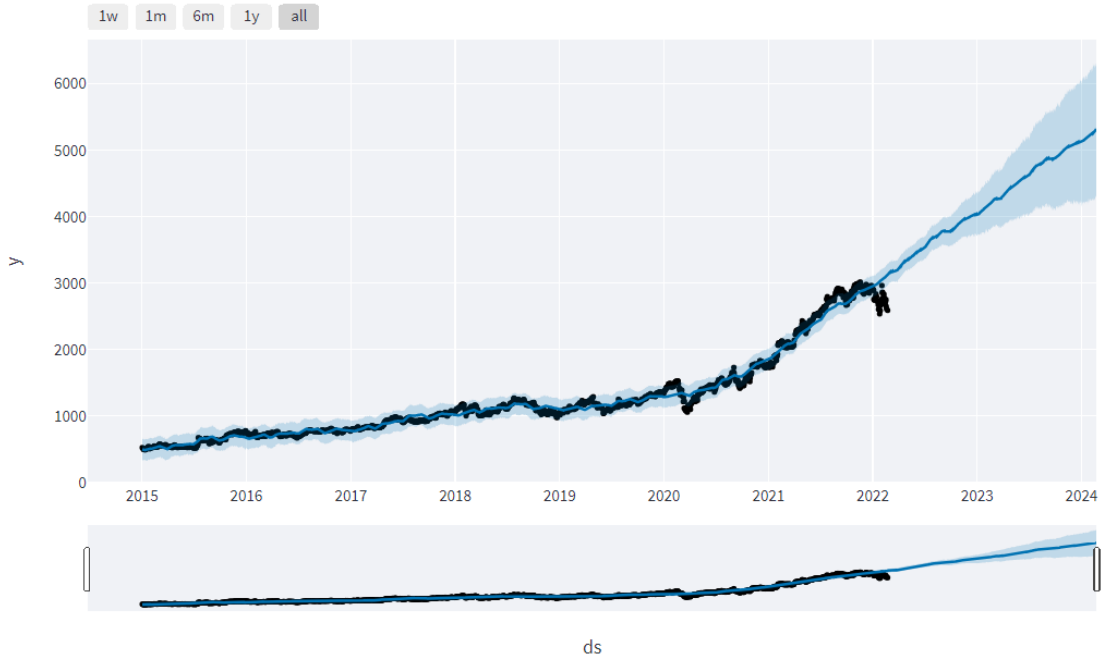
Las **dependencias** necesarias: requirements.txt

Enlaces a la aplicación : https://share.streamlit.io/sebasespon/api-forecast-stocks/main/main.py

Enlace a Video Tutorial Youtube:

A continuación, adjuntamos algunas imágenes de la aplicación.

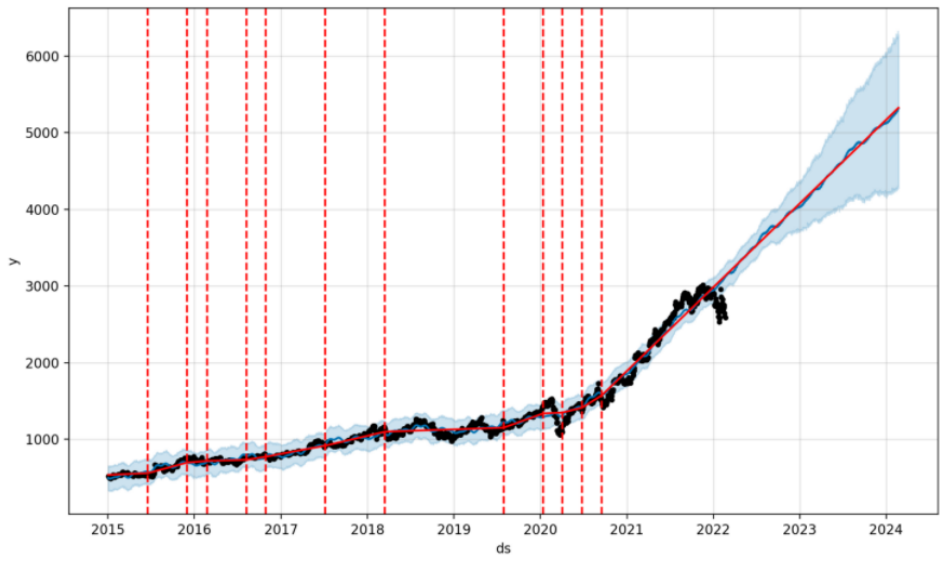


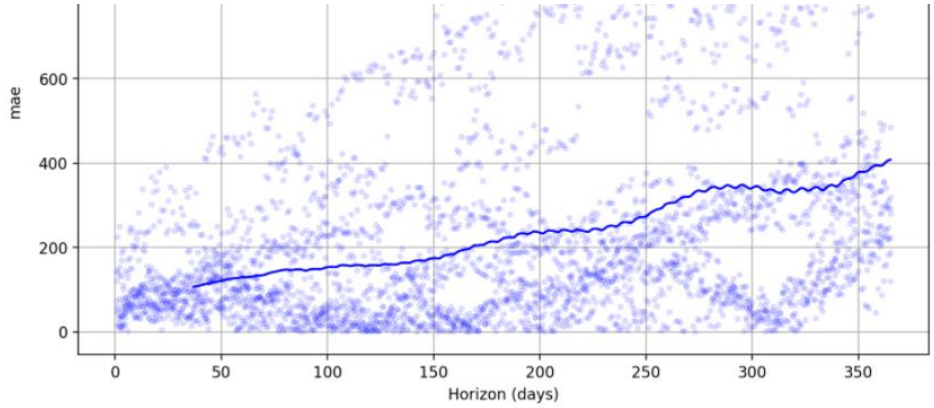


ChangePoints Plot

Los **Changepoints** son los puntos de fecha en los que las series temporales presentan cambios bruscos en la trayectoria.

Por defecto, Prophet añade **25 puntos** de cambio al 80% inicial del conjunto de datos.





Authors

Sebastian Esponda 😎

Gary Martin 😉

Levi Vilchez 😛

Javier Jimenez Peña 😆

Youssef Ouabi😁

8. Bibliografía

Barria, C. (6 de diciembre de 2021). 6 errores comunes que comete la gente al invertir su dinero en la bolsa y cómo evitarlos. BBC News Mundo. https://www.bbc.com/mundo/noticias-59470575

BBVA. (s.f.). Inversión con Big Data. https://www.bbva.es/finanzas-vistazo/ef/fondos-inversion/inversion-big-data.html

Blad, D. (3 de noviembre de 2020). yfinance Library – A Complete Guide. algotrading101. https://algotrading101.com/learn/yfinance-guide/

Digital Guide IONOS. (s.f.). ¿Qué es Keras?. https://www.ionos.es/digitalguide/online-marketing/marketing-para-motores-de-busqueda/que-es-keras/

Letham, B. & Taylor, S. (23 de febrero 2017). Prophet: forecasting at scale. https://research.facebook.com/blog/2017/02/prophet-forecasting-at-scale/

Neural Prophet. (s.f) Quick start guide. https://neuralprophet.com/html/index.html

Programador Clic. (s.f.). plotly ---- Biblioteca python de dibujo interactivo más simple y hermosa que matplotlib. https://programmerclick.com/article/5132768509/

PyCaret. (s.f.). Welcome to PyCaret. https://pycaret.gitbook.io/docs/

Predicting Stock Prices :https://www.youtube.com/watch?v=PuZY9q-aKLw

Financial Data with Yfinance: : https://www.youtube.com/watch?v=7wAQCwdvqqo

Financial Dashboards with Plotly: https://www.youtube.com/watch?v=OJNxE1FjtXU

Documentación Plotly: https://plotly.com/python/plotly-fundamentals

Articulos Plotly: https://www.geeksforgeeks.org/python-plotly-tutorial/

Prophet Documentation: https://facebook.github.io/prophet/docs/quick_start.html

Time Series FbProphet: https://www.youtube.com/watch?v=VtItg-J6-CI

Kaggle Notebook: https://www.kaggle.com/robikscube/time-series-forecasting-with-prophet

Video Tutorial Time Series Pycaret: https://www.youtube.com/watch?v=juowHZNHADM

Functions Pycaret Video: https://www.youtube.com/watch?v=lHBAHggc4Jg

Documentación Oficial Pycaret: https://github.com/pycaret/pycaret/blob/time_series/time_series_101.ipynb

Neural Prophet Documentation: https://neuralprophet.com/html/index.html

Forecasting Weather with Neural Prophet: https://www.youtube.com/watch?v=mgX0Iz4q0bE

GitHub Tutorial: https://github.com/ourownstory/neural_prophet

Articulo Towards Data Science Time series: https://towardsdatascience.com/time-series-forecasting-with-pycaret-regression-module-237b703a0c63#:~:text=PyCaret%20Regression%20Module%20is%20a,%2C%20or%20'predictors'

Pycaret Notebook Tutorial:

https://www.pycaret.org/tutorials/html/REG101.html

Pycaret Regression Tutorial: https://github.com/pycaret/pycaret/blob/master/tutorials/Regression%20Tutorial%20Level%20Intermediate%20-%20REG102.ipynb

Artículo section.io: https://www.section.io/engineering-education/stock-price-prediction-using-python/

Tutorial LSTM Github: https://github.com/ahmadmardeni1/Stock-price-predection-using-Python

Machine Learning Mistery: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

DataFlair Project: https://data-flair.training/blogs/stock-price-prediction-machine-learning-project-in-python/