INTELIGENCIA Y PROCESOS DE NEGOCIOS EN LA ECONOMÍA DIGITAL

Lic. y Prof. Antonio Sottile Bordallo
Profesor Titular Tecnología de la Información I y II
MSC, Lic. y Prof. Daniel Guillermo Cavaller Riva
Profesor Asociado Tecnología de la Información I y II
Lic. Emiliano Andrés Dueñas
Profesor Adjunto Tecnología de la Información I y II
Cdor. Cristian Darío Ortega Yubro
Jefe de Trabajos Prácticos Tecnología de la Información I y II
Lic. Héctor Nicolás Sosa
Profesional Adscripto Tecnología de la Información I y II

{antonio.sottile; daniel.cavaller; cristian.ortega; hector.sosa; emiliano.duenas}@fce.uncu.edu.ar
http://fce.uncu.edu.ar
Google Colab: https://github.com/Inteligencia-y-Procesos-de-Negocios-en-la-Economia-Digital

Introducción

Conveniente es lograr visualizar las oportunidades de los nuevos modelos de negocios basados en avances tales como las criptodivisas, la criptografía, la inteligencia artificial en todo su esplendor, que avanza día a día, el internet de las cosas y en especial la analítica de datos que, ponen de manifiesto los nuevos desafíos emergentes de la economía digital lo que conlleva el aprovechamiento al máximo de las tecnologías de la información y de la comunicación, y pone de manifiesto la necesidad de conocerlas para interactuar con ellas.

Algunas de las plataformas facilitadoras de la economía digital, que forman parte del ecosistema digital, son las que se detallan a continuación:

- la computación en la nube,
- las redes sociales, y
- la analítica de datos.

Los datos generan una enorme cantidad de información que, procesada mediante herramientas analíticas en la nube, constituyen un insumo indispensable para el diseño de estrategias productivas y de mercado a través del conocimiento, pero también es un insumo por ejemplo, para el control y el desarrollo de auditorías continuas, para el análisis financiero, para la detección de fraude, etc., tareas que pueden automatizarse, a través de la generación de modelos de aprendizaje supervisado o no supervisado, y si es supervisado en este caso, con la supervisión de los profesionales de las ciencias económicas.

Los procesos de negocios de un ecosistema digital incluyen, por un lado, aplicaciones horizontales como las de los servicios financieros, los servicios contables y los de recursos humanos y por otro lado, las aplicaciones verticales asociadas a aquellas actividades que son específicas, como por ejemplo a la actividad financiera, al sector público, al sector manufacturero, el comercio electrónico, las telecomunicaciones, el transporte y la salud entre otras actividades (Parrilla, 2018).

Los procesos analíticos del conocimiento, es decir aquellos procesos incluidos en la inteligencia artificial, se refieren a actividades de alta especialización y complejidad, destacándose entre ellas, los servicios de analítica de datos de negocios, los servicios de investigación económica, y al desarrollo tecnológico necesario y aplicado a los negocios de la economía digital, todo ello mediante el uso de lenguajes de alto nivel, como lo es Python. El análisis de datos de negocios o de activos permite hacer más y mejores pronósticos a través de los análisis predictivos con el uso de los algoritmos correspondientes, así como también ajustar decisiones con base en información completa y en tiempo real, siendo la información el insumo del conocimiento.

En esta instancia, es oportuno incorporar el concepto de activo digital. Un activo digital (*Negocios + Digitales*, s. f.) puede adquirir valor de mercado y generar ingresos de manera genuina, e incluso aumentar el valor de una marca. La percepción del valor económico que puede adquirir un bien intangible depende de múltiples factores

determinados, por ejemplo: la visión de quien toma las decisiones, o la capacidad de inversión en activos digitales, o la aplicación de adecuadas plataformas digitales, para el desarrollo de la economía digital. Todo ello, y mucho más significa trabajar el activo digital y su ecosistema digital correspondiente, como un activo intangible. Por lo tanto, el concepto del activo digital está basado en el diseño de estrategias y plataformas que faciliten la conexión entre la marca y los usuarios en un proceso de negocio, lo que favorece la relación entre las partes intervinientes para la obtención de beneficios mutuos, lo que impacta directamente en los negocios de la economía digital a través de nuevos procesos, la inteligencia de negocios aplicada a esos procesos y la gestión de las finanzas de los activos digitales.

La actividad financiera es la actividad que constituye una de las actividades principales de la economía digital, con nuevos procesos de negocios como por ejemplo aquellos destinados a la economía colaborativa. Por lo tanto, la actividad financiera se encuentra inmersa en una profunda transformación, con condicionantes tantos internos como externos (Raquel, Ángel, & Rodrigo, 2019).

Por lo expuesto anteriormente se manifiestan múltiples necesidades de formación de los profesionales de las ciencias económicas con el objetivo de prepararlos con una mirada digitalizada de la economía y en especial de las finanzas, siendo la actividad financiera transversal a cada uno de los profesionales, es decir: al licenciado en administración, al contador público y al licenciado en economía, con lo cual todos tienen injerencia.

Sin los profesionales en ciencias económicas con destrezas digitales para la innovación y el trabajo técnico en las distintas organizaciones a través de la analítica de datos, no se podrá aprovechar las oportunidades de la economía digital («Políticas 4.0 para la cuarta revolución industrial», 2018). Las brechas entre las necesidades de las organizaciones y la oferta de estos profesionales son cada vez más grandes y notorias. La formación universitaria no puede permanecer al margen de estas transformaciones. Asimismo, el cambio de mentalidad que impulsará una transformación real de procesos de negocio en la economía digital y puesta en valor de datos de nuestras organizaciones tiene que venir impulsado por los profesionales de las ciencias económicas que ven la necesidad de incursionar en el campo de la inteligencia y procesos de negocios acordes a los avances tecnológicos.

Objetivos

- Evidenciar servicios facilitados por las tecnologías de la información y de la comunicación, en los que se destacan la industria de procesos analíticos o de conocimiento aplicados.
- Evidenciar el grado de absorción de las aplicaciones digitales mediante su demanda por servicios y aplicaciones en la economía digital.

Ejes de la propuesta

Ejes centrales

- Esbozar modelos de aprendizaje profundo a datos de activos pertenecientes a los ecosistemas de la economía digital, puntualmente a las aplicaciones verticales asociadas a la actividad financiera.
- Aplicar modelos generados con aprendizaje profundo, con el fin del control de datos provenientes de la economía digital.
- Estudiar algoritmos y aplicaciones utilizables.

Ejes periféricos

- Poner a disposición de la Facultad de Ciencias Económicas – Sede Central y Delegación San Rafael – de la Universidad Nacional de Cuyo, los resultados y conclusiones obtenidas en la presente investigación, para fomentar el estudio continuo y profundizado de la Ciencia de Datos y del lenguaje Python, revelando su importancia para el profesional de Ciencias Económicas.
- Publicar los resultados obtenidos y las conclusiones arribadas de la presente investigación en las Jornadas Provinciales de Ciencias Económicas, en el Congreso Nacional de Ciencias Económicas, en las

Jornadas Argentinas de Informática e Investigación Operativa, y en el repositorio digital del Sistema Integrado de Documentación de la Universidad Nacional de Cuyo.

Marco teórico

En el ecosistema digital, que incluye el ecosistema digital financiero, hay varias áreas en las que la inteligencia y los procesos de negocios pueden desempeñar un papel muy importante. Una de ellas es la detección de anomalías o datos atípicos para optimizar los costos de una organización o para identificar oportunidades de negocios basándose exclusivamente en los principios del descubrimiento de patrones de los datos bajo análisis (Souiden, Brahmi, & Lafi, 2017).

Hoy en día existe la posibilidad de encontrar patrones complejos en los datos que están en línea, disponibles para todos los actores, no solo las instituciones financieras que forman parte del ecosistema digital, y conocer sobre esos datos el respaldo que otorgan a las actividades operativas como las actividades de inversión de las distintas organizaciones (*Digital Economy. Emerging Technologies and Business Innovation*, s. f.).

Ese servicio analítico, es un valor extra que adquieren los profesionales de las ciencias económicas que se especializan en la analítica de datos, más comúnmente denominada Ciencia de Datos. Por ejemplo, una parte de la inteligencia artificial colabora en la automatización de los procesos de negocios financieros contribuyendo a la detección de fraude, que no son más que ilícitos y/o hechos de corrupción, también contribuye a la validación de diferentes tipos de transacciones y su correcta registración velando por la calidad óptima del dato registrado para la correcta modelización. Una preocupación constante en la actividad financiera es la relativa a la detección y prevención del fraude. Tener una gran cantidad cada vez mayor de datos confidenciales en línea y que puedan ser accedidos a través de Internet significa que el riesgo de fuga de esos datos hoy en día es mayor que antes. Esto representa un riesgo importante para la seguridad de los datos, y un desafío para la analítica de dato y la confección de algoritmos a medida de las distintas organizaciones.

Antes, en el pasado, la detección de fraudes se basaba en un conjunto complejo de reglas. Hoy el enfoque moderno apunta a identificar anomalías de los datos, o bases de datos anómalas, con un comportamiento de los datos o en los patrones detectados que revelen amenazas potenciales y reales, alertando así al equipo de profesionales para ayudarlos a prevenir o mitigar los riesgos inherentes de la inclusión de la organización al ecosistema digital. El desafío es construir modelos que puedan identificar amenazas con precisión, pero al mismo tiempo minimicen las situaciones de falsos positivos en las que se genera alerta por error. La flexibilidad de los sistemas inteligentes y la potencialidad de los lenguajes de programación como Python más que adecuada (Troiano, Bhandari, & Villa, 2020).

En ese contexto, uno de los desafíos de seguridad de la inteligencia y procesos de negocios en la economía digital son los relacionados con el de las plataformas abiertas como las de open banking, porque se está impulsando a la integración total de las mismas a través de las interfaces de programación de aplicaciones (API), en donde las transacciones digitales son efectuadas entre distintos dispositivos, sin la intervención de las personas humanas. En este punto es donde cobran importancia los distintos algoritmos de inteligencia de negocios que son aplicables a esos procesos de negocios automatizados.

Sin embargo, el área principal de interés de la inteligencia de negocios tiene que ver con la aplicación del aprendizaje profundo para la gestión y el comercio de activos digitales. Desde el año 1970, la automatización y el apoyo para la toma de decisiones en las distintas organizaciones del sector financiero ha visto una creciente inversión en tecnología de información y comunicación. Esto se debe a los mayores beneficios de gestionar riesgos de forma cuantitativa (Troiano et al., 2020). Los modelos generados y utilizados por las organizaciones financieras son secretos, ya que incrementan el valor estratégico para los participantes del ecosistema digital. El propósito de la aplicación de estos algoritmos es descubrir patrones que puedan predecir movimientos de precios y tendencias del mercado de la economía digital, obteniendo así ganancias al administrar estos activos y comercializarlos digitalmente en el mercado. En una economía no digital, la actividad financiera se basa en datos estructurados que son representados por precios y volúmenes para análisis estadístico, dejando datos no estructurados como las noticias en línea, las imágenes satelitales, los comentarios en redes sociales, al juicio y valoración humana, perdiendo así una ventaja competitiva frente a otros actores que ejercen la actividad financiera. Pero, los avances en el procesamiento del lenguaje natural gracias a la inteligencia artificial está permitiendo la detección automática de noticias y redes sociales para anticipar los movimientos del mercado y de los consumidores, o usuarios basados en el análisis de los datos no estructurados.

El análisis de expresión y el análisis de sentimientos son algunas de las herramientas más avanzadas que se utilizan para comprender el estado de ánimo del mercado en tiempo real ayudando a producir un gran avance de la inteligencia de negocios y de los procesos de negocios en el dominio financiero (Raquel et al., 2019).

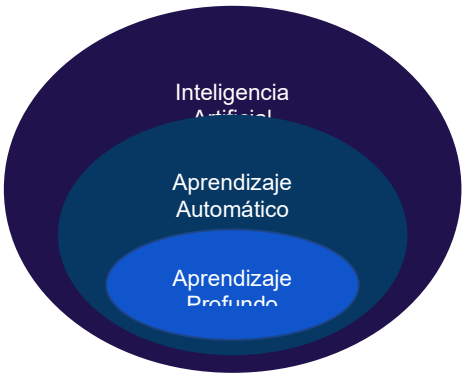
Se desarrollará a continuación un ejemplo de análisis de datos de un activo financiero como las acciones, con el objetivo de detectar patrones y realizar modelos predictivos a través del aprendizaje profundo, que es ni más ni menos que desarrollar inteligencia artificial, usando algoritmos de redes neuronales, implementados con TensorFlow, dejando de manifiesto lo que un profesional de ciencias económicas podría lograr contando solamente con un dispositivo, conexión a internet y conocimientos previos de Ciencias de Datos, partiendo de lo básico hasta llegar a un nivel intermedio, sin ahondar en explicaciones de conceptos netamente estadísticos.

Estado del arte

Las finanzas cuantitativas y la inteligencia artificial compartieron un marco teórico común hace mucho tiempo, antes de que fueran reconocidas como campos de investigación independientes: ese marco teórico común fue el estudio de la probabilidad, y aún lo es. La probabilidad hace posible estimar la hipótesis sobre la base de la evidencia que se ha observado y permite modelar incluso los vínculos más complejos entre las variables que conforman un modelo de realidad. La noción de probabilidad es muy antigua. Fue en el siglo XVII que se obtuvo la primera formulación matemática. Se dice que el concepto de probabilidad surgió en el año 1654 partiendo de una pregunta sobre apuestas, formulada por dos famosos matemáticos franceses: Blaise Pascal y Pierre de Fermat.

Las actividad financiera fue una de las primeras actividades en obtener ganancias gracias al progreso realizado en la informática, durante la Segunda Guerra Mundial. La informática atrajo al dominio financiero debido al potencial vislumbrado de la automatización en el análisis y procesamiento de los datos para desarrollar la actividad bancaria y para desarrollar los servicios de contabilidad a una velocidad que en ese entonces era imposible de igualar por los humanos. Aunque, en ese momento, el enfoque estaba en la adopción de métodos automatizados tempranos para procesar datos comerciales conocidos tales métodos como procesamiento electrónico de datos, la inteligencia de negocios y los procesos de negocios estaban logrando sus primeros éxitos mediante el desarrollo de los sistemas expertos.

Imagen Nº 1 - Inteligencia artificial, aprendizaje automático y aprendizaje profundo



Metodología del Aprendizaje Profundo

La metodología del aprendizaje profundo para la economía digital, en el ámbito financiero aplicado al análisis de datos, puede desarrollarse en una representación de esos datos en una series de tiempo en un espacio de dimensiones reducidas. Esto nos permite tener en cuenta un mercado en general, de acuerdo a la actividad bajo análisis. La aplicación de un algoritmo específico para lograr ese análisis puede efectuarse con redes neuronales, implementadas en el presente caso con Keras y TensorFlow. Las redes neuronales han asumido un papel central gracias a su versatilidad en la construcción de modelos complejos impulsados por los datos, para resolver una

pluralidad de problemas como el análisis predictivo, los precios, la asignación de activos digitales y otros problemas (Troiano et al., 2020).

Metodología del estudio

Una de las fuentes más populares de datos financieros gratuitos es Yahoo Finance. Contiene no solo los precios de las acciones históricas y actuales en diferentes frecuencias (diarias, semanales, mensuales), sino también métricas calculadas, como la beta (una medida de la volatilidad de un activo individual en comparación con la volatilidad de todo el mercado) y mucho más. En este estudio, nos centramos en recuperar los precios históricos de las acciones de YPF, con unas simples instrucciones de Python con la herramienta Google Colab.

Colab es una herramienta que permite ejecutar y programar Python desde cualquier dispositivo que se esté utilizando, con la posibilidad de utilizar los recursos computacionales que Google pone a disposición, y el atractivo de la integralidad con TensorFlow. De acuerdo a la presentación que hace Colab, la aplicación puede facilitar el trabajo ya sea de un alumno, un científico de datos o un investigador de inteligencia artificial («Google Colaboratory», s. f.) y de esa forma desarrollar modelos. Por ejemplo, una de las librerías preinstaladas en Google Colab es Pandas, algunos conocen esta librería como aquella que se utiliza para ciertos cálculos matemáticos, pero la realidad es que esta librería nos ofrece una facilidad para trabajar con estructuras de datos y pone a disposición herramientas para análisis de datos. Pandas se utiliza para investigaciones académicas incluyendo tópicos de las finanzas, la economía, la estadística y otros. El nombre Pandas es un derivado de dos palabras Panel Data, conocido en el campo de la econometría para el análisis de datos multidimensionales. Los archivos de ejemplo generados en Google Colab se pueden alojar en los repositorios de GitHub. El enlace del repositorio es el siguiente:

https://github.com/cristiandarioortegayubro/Inteligencia-y-Procesos-de-Negocios-en-la-Economia-Digital

Una vez ingresado a GitHub, clonar los archivos, y tener instalado Python. Con la aplicación práctica de la presente investigación a través de Google Colab y lenguaje Python, se alcanzan los dos objetivos generales.

- Evidenciar servicios facilitados por las tecnologías de la información y de la comunicación, y
- Evidenciar el grado de absorción de las aplicaciones digitales mediante su demanda por servicios y aplicaciones en la economía digital.

Obtención de datos financieros

Primero se importan las librerías necesarias para la recolección de los datos en línea financieros para analizar:

```
import pandas as pd
import pandas_datareader as data
import datetime as dt
import matplotlib.pyplot as plot
```

De esa forma se preparan las librerías necesarias para la obtención de los datos que en esta oportunidad será a través de datareader de pandas («Pandas datareader documentation», s. f.). A continuación se selecciona y se define el conjunto de acciones para obtener su cotización y el período que se requiere:

```
YPF = data.DataReader ("YPF", start='2019-06-01', end='2020-04-12', data_source='yahoo')
```

La sintaxis es bastante simple hasta ahora, se crea la variable YPF que obtiene los datos de la API datareader de pandas, comenzando el día 01 de junio del 2019 hasta el día 12 de abril del 2020. La fuente de esos datos son los consignados en línea en yahoo. De esta forma, y con ese simple comando, no hay que visitar ninguna plataforma digital para la obtención de estos datos financieros necesarios para la construcción del modelo de aprendizaje profundo. Entonces, dentro de la variable YPF están los datos requeridos de yahoo, pero a esos datos se les debe dar un formato de tabla para trabajar con ellos, y poder visualizarlos de la forma en la que estamos acostumbrados, utilizando para ello la librería pandas.

```
ypf = pd.DataFrame (YPF)
```

Se crea una nueva variable denominada esta vez ypf (en minúsculas) para generar una tabla con los valores de la variable YPF creada anteriormente, y se visualiza los valores a través del comando head() que muestra los primeros cinco registros, con la siguiente sintaxis:

ypf.head()

Imágen Nº 2 - Dataframe generado con los valores de las acciones de YPF

```
ypf.head()
```

Date	High	Low	Open	Close	Volume	Adj Close
2019-06-03	15.24	14.55	14.94	14.97	1471900.0	14.854709
2019-06-04	15.22	14.71	15.05	15.13	2318800.0	15.013476
2019-06-05	15.28	14.89	15.03	15.06	947700.0	14.944016
2019-06-06	15.40	14.88	15.20	15.23	868200.0	15.112705
2019-06-07	15.69	15.19	15.33	15.64	1189200.0	15.519548

Lo único que quedaría saber es cuántos valores observados tiene el dataframe y cuantas variables considera. Esa información se obtiene con siguiente comando:

ypf.shape

El resultado se indica de la siguiente forma (217, 6), es decir, 217 observaciones que corresponden a los días en los que cotizaron las acciones de YPF y 6 son las columnas. A primera vista pareciera que no estarían las cotizaciones de los días sábados y domingos. Con unos simples comandos se puede constatar tal apreciación. Lo interesante de la tabla de la Imágen Nº 2 es que el nombre de las filas corresponde a la fecha de cotización del período bajo análisis. Ese detalle se puede adecuar de acuerdo al objetivo establecido para el análisis de los datos a través de la limpieza y transformación de los datos financieros.

Limpieza y transformación de los datos financieros

Para poder limpiar la tabla generada y crear otra columna denominada fecha (Date) se puede utilizar la siguiente metodología, primero crear una variable Date para cargar en ella todos los valores de los nombres de cada fila, es decir cada fecha.

Date = ypf.index.values

Luego se inserta como columna en la tabla, en la primera posición con el siguiente comando:

ypf.insert (0,column='Date',value=Date)

Es decir, a la tabla ypf se le inserta una columna en la posición cero, denominada Date, tomando los valores de la variable Date creada anteriormente.

Imagen Nº 3 - Generando columna Date

```
print(ypf)
```

```
                 Date   High    Low   Open  Close     Volume  Adj Close
Date
2019-06-03 2019-06-03  15.24  14.55  14.94  14.97  1471900.0  14.854709
2019-06-04 2019-06-04  15.22  14.71  15.05  15.13  2318800.0  15.013476
2019-06-05 2019-06-05  15.28  14.89  15.03  15.06   947700.0  14.944016
2019-06-06 2019-06-06  15.40  14.88  15.20  15.23   868200.0  15.112705
2019-06-07 2019-06-07  15.69  15.19  15.33  15.64  1189200.0  15.519548
...                ...    ...    ...    ...    ...        ...        ...
2020-04-03 2020-04-03   4.60   4.15   4.56   4.36  1236900.0   4.360000
2020-04-06 2020-04-06   4.55   4.23   4.30   4.26  1585000.0   4.260000
2020-04-07 2020-04-07   4.58   4.01   4.40   4.13  2552500.0   4.130000
2020-04-08 2020-04-08   4.33   4.08   4.13   4.23  1490700.0   4.230000
2020-04-09 2020-04-09   4.48   4.05   4.39   4.20  1673300.0   4.200000

[217 rows x 7 columns]
```

Ahora la tabla posee 217 filas y 7 columnas, porque se agregó la columna Date, pero se debe arreglar el nombre de las filas, para que no se dupliquen las fechas en el dataframe, lo que se consigue con el siguiente comando:

ypf.reset_index (drop=True, inplace=True)

Imágen Nº 4 - Arreglando los nombres de las filas

```
print(ypf)
```

```
           Date   High    Low   Open  Close     Volume  Adj Close
0    2019-06-03  15.24  14.55  14.94  14.97  1471900.0  14.854709
1    2019-06-04  15.22  14.71  15.05  15.13  2318800.0  15.013476
2    2019-06-05  15.28  14.89  15.03  15.06   947700.0  14.944016
3    2019-06-06  15.40  14.88  15.20  15.23   868200.0  15.112705
4    2019-06-07  15.69  15.19  15.33  15.64  1189200.0  15.519548
..          ...    ...    ...    ...    ...        ...        ...
212  2020-04-03   4.60   4.15   4.56   4.36  1236900.0   4.360000
213  2020-04-06   4.55   4.23   4.30   4.26  1585000.0   4.260000
214  2020-04-07   4.58   4.01   4.40   4.13  2552500.0   4.130000
215  2020-04-08   4.33   4.08   4.13   4.23  1490700.0   4.230000
216  2020-04-09   4.48   4.05   4.39   4.20  1673300.0   4.200000

[217 rows x 7 columns]
```

Ahora se creará una columna nueva con los nombres de los días correspondiente al valor consignado en el campo Date, con el siguiente comando:

ypf['Day'] = pd.DatetimeIndex (ypf['Date']).day_name()

Ahora la tabla quedó de la siguiente forma:

Imagen Nº 5 - Nombre del día según fecha de cotización

```
print(ypf)
```

```
           Date   High    Low   Open  Close     Volume  Adj Close        Day
0    2019-06-03  15.24  14.55  14.94  14.97  1471900.0  14.854709     Monday
1    2019-06-04  15.22  14.71  15.05  15.13  2318800.0  15.013476    Tuesday
2    2019-06-05  15.28  14.89  15.03  15.06   947700.0  14.944016  Wednesday
3    2019-06-06  15.40  14.88  15.20  15.23   868200.0  15.112705   Thursday
4    2019-06-07  15.69  15.19  15.33  15.64  1189200.0  15.519548     Friday
..          ...    ...    ...    ...    ...        ...        ...        ...
212  2020-04-03   4.60   4.15   4.56   4.36  1236900.0   4.360000     Friday
213  2020-04-06   4.55   4.23   4.30   4.26  1585000.0   4.260000     Monday
214  2020-04-07   4.58   4.01   4.40   4.13  2552500.0   4.130000    Tuesday
215  2020-04-08   4.33   4.08   4.13   4.23  1490700.0   4.230000  Wednesday
216  2020-04-09   4.48   4.05   4.39   4.20  1673300.0   4.200000   Thursday

[217 rows x 8 columns]
```

La tabla tiene 217 observaciones (filas) y ahora, 8 columnas.

Ahora bien, los precios de los activos generalmente no son estacionarios, y cambian con el tiempo, lo que puede constatarse con el estudio del valor de la media y la varianza. Para intentar lograr una estacionalidad en el precio, se podría calcular los retornos simples de los activos de la siguiente forma:

$$R_n = \frac{P_n - P_{n-1}}{P_{n-1}}$$

donde $R_n$es el retorno del día n. El cálculo del retorno entonces, se hará con los datos de la columna de cierre ajustado (Adj Close), para ello, se genera la variable CloseAdj de la siguiente forma:

CloseAdj = ypf.loc[:, ['Adj Close']]

A continuación se cambia el nombre de la columna en el nuevo dataframe CloseAdj, con el objetivo de no generar un error en la sintaxis para el posterior cálculo de los retornos simples:

CloseAdj.rename(columns={'Adj Close':'Adj_Close'}, inplace=True)

Y se calcula el retorno simple, agregando la pertinente columna al dataframe CloseAdj, con el siguiente comando:

CloseAdj['Retorno simple'] = CloseAdj.Adj_Close.pct_change()

Imagen Nº 6 - Retorno Simple

```
print(CloseAdj)
```

```
     Adj_Close
0    14.854709
1    15.013476
2    14.944016
3    15.112705
4    15.519548
..         ...
212   4.360000
213   4.260000
214   4.130000
215   4.230000
216   4.200000

[217 rows x 1 columns]
```

El primer dato de la fila Retorno simple contiene un valor NaN (no es un número) ya que no existe el precio n-1 para esa posición. Pero al dataframe debería agregarse el campo Date.

CloseAdj.insert (0,column='Date',value=Date)

Imágen Nº 7 - Retorno Simple con fechas

```
print(CloseAdj)
```

```
          Date  Adj_Close  Retorno_simple
0   2019-06-03  14.854709             NaN
1   2019-06-04  15.013476        0.010688
2   2019-06-05  14.944016       -0.004627
3   2019-06-06  15.112705        0.011288
4   2019-06-07  15.519548        0.026921
..         ...        ...             ...
212 2020-04-03   4.360000        0.000000
213 2020-04-06   4.260000       -0.022936
214 2020-04-07   4.130000       -0.030516
215 2020-04-08   4.230000        0.024213
216 2020-04-09   4.200000       -0.007092

[217 rows x 3 columns]
```

Visualización de los datos financieros

La visualización de los datos financieros bajo análisis es muy importante. Permite proyectar conclusiones previas del desarrollo del modelo de aprendizaje. Por lo tanto, se debe preparar y activar las librerías que se utilizan para desarrollar los gráficos, y se hace de la siguiente forma:

```
import matplotlib as plot
import matplotlib.pyplot as plt
```

La idea de generar los dos alias, plot y plt es porque cada uno de ellos contiene diferentes sintaxis en lo que corresponde al ajuste de los parámetros de los gráficos que se desarrollen. Además, los gráficos poseen estilos que son como plantillas predeterminadas que les asigna color y tipografía específica. Para conocer los estilos disponibles de los gráficos, se ejecuta el siguiente comando:

```
print (plot.style.available)
```

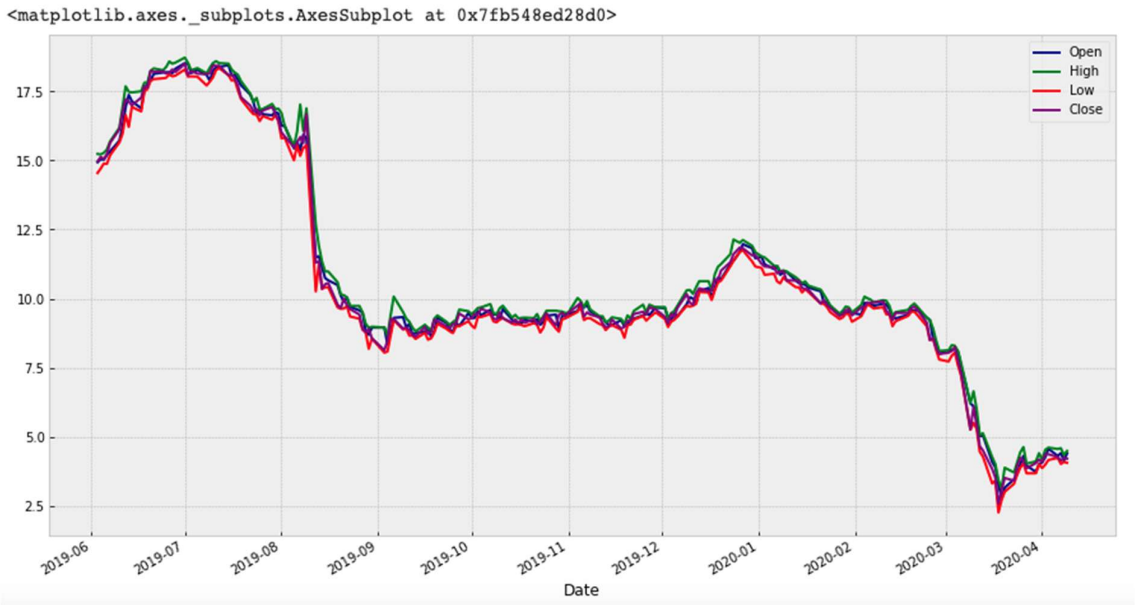
El resultado de la ejecución de esa celda de código evidenciará los distintos estilos disponibles. A continuación, el resultado generado:

['Solarize_Light2', '_classic_test_patch', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn', 'seaborn-bright', 'seaborn-colorblind', 'seaborn-dark', 'seaborn-dark-palette', 'seaborn-darkgrid', 'seaborn-deep', 'seaborn-muted', 'seaborn-notebook', 'seaborn-paper', 'seaborn-pastel', 'seaborn-poster', 'seaborn-talk', 'seaborn-ticks', 'seaborn-white', 'seaborn-whitegrid', 'tableau-colorblind10']

Rápidamente se puede hacer un gráfico lineal básico, ejecutando el siguiente comando:

```
plot.style.use('bmh')
ypf.plot('Date',['Open','High','Low','Close'], kind='line', figsize=(15,8), color=['darkblue','green','red','purple'])
```

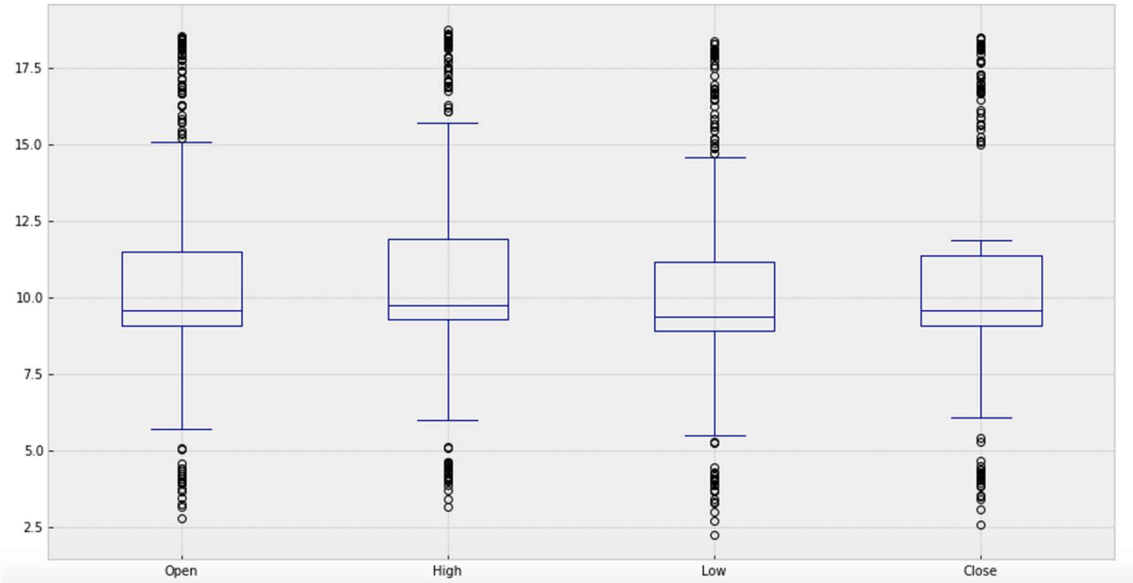
Imágen Nº 8 - Gráfico lineal



o un gráfico de cajas...

Imágen Nº 9 - Gráfico de cajas

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb548d99dd8>
```



Estos gráficos son muy sencillos de realizar, y nos permiten obtener información acerca de los datos analizados.

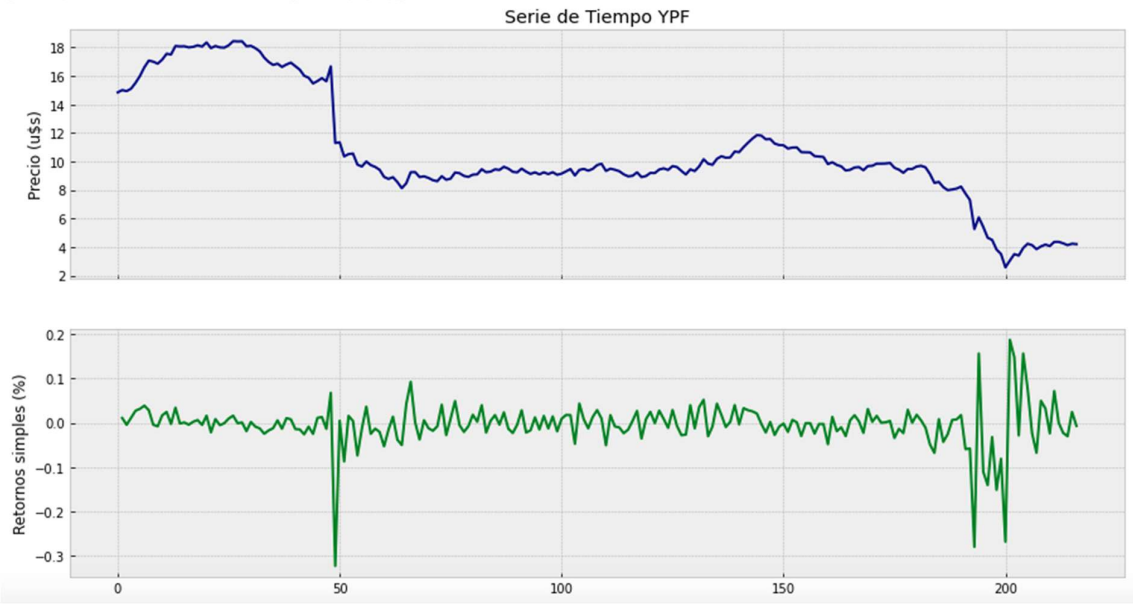
Visualización serie de tiempo de los datos financieros

En esta instancia se va a graficar la serie de tiempo del precio de las acciones de YPF en conjunto con su retorno diario. Para ello se ejecuta el siguiente comando:

```
plot.style.use('bmh')
fig, ax = plot.subplots(2, 1, figsize=(15,8), sharex=True)
CloseAdj.Adj_Close.plot(ax=ax[0], color=['darkblue'])
ax[0].set(title = 'Serie de Tiempo YPF', ylabel = 'Precio (u$s)')
CloseAdj.Retorno_simple.plot(ax=ax[1], color=['green'])
ax[1].set(ylabel = 'Retornos simples (%)')
```

Imágen Nº 10 - Serie de Tiempo

```
[Text(0, 0.5, 'Retornos simples (%)')]
```



Pero en esta instancia se puede desarrollar un modelo alternativo para la visualización de las series de tiempo aditivo en el que una serie de tiempo se representa como una combinación de patrones en diferentes escalas de tiempo (diario, semanal, mensual, anual, etc.) junto con la tendencia general (Troiano et al., 2020). Primero se activan las librerías necesarias:

```
import seaborn as sns
from fbprophet import Prophet
```

Una ventaja práctica de usar la biblioteca Prophet es que se puede pronosticar valores futuros de las series de tiempo, junto con un intervalo de confianza que indica el nivel de incertidumbre. En nuestro caso se creará un pronóstico de tres meses a futuro.

Previo a ello, debe ajustarse el dataframe:

```
print(CloseAdj)
CloseAdj.rename(columns={'Date': 'ds', 'Adj_Close': 'y'}, inplace=True)
CloseAdj = CloseAdj.drop(columns=['Retorno_simple'])
print(CloseAdj)
```

Por ese motivo, se cambian los nombres de las columnas, y posteriormente a ello se divide la serie de tiempo en los conjuntos de datos de entrenamiento y de prueba, de la siguiente manera:

```
train_indices = CloseAdj.ds.apply(lambda x: x.year) < 2021
df_train = CloseAdj.loc[train_indices].dropna()
df_test = CloseAdj.loc[~train_indices].reset_index(drop=True)
```

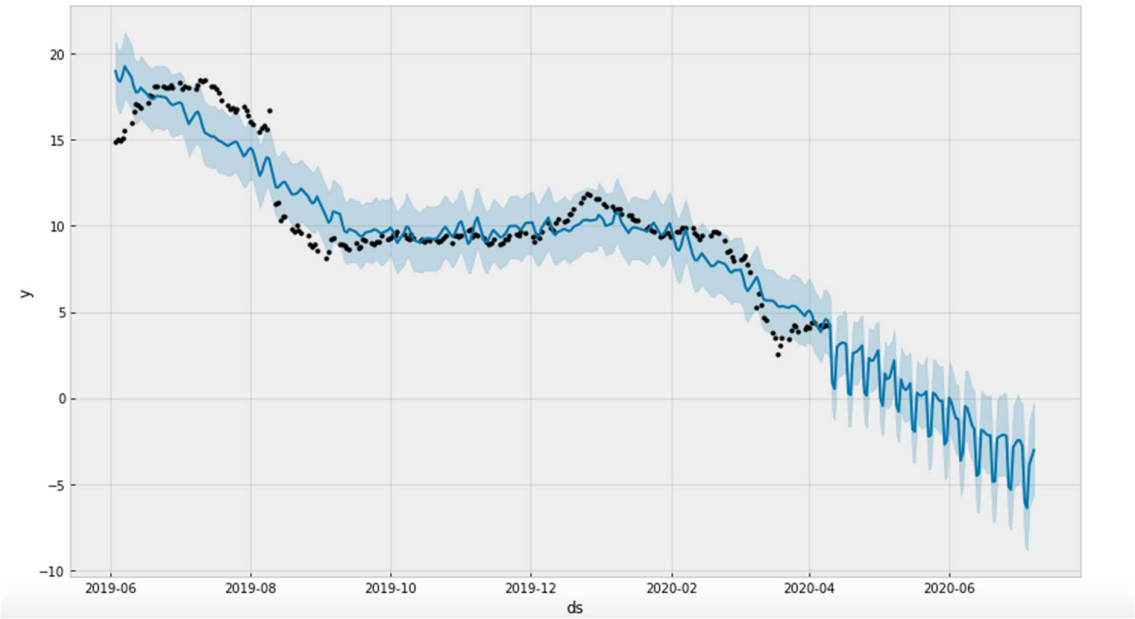
Luego se crea la instancia del modelo y se ajusta a los datos bajo análisis:

```
model_prophet = Prophet(seasonality_mode='additive')
model_prophet.add_seasonality(name='monthly', period=30.5, fourier_order=5)
model_prophet.fit(df_train)
```

Y se pronostica los precios de las acciones de YPF, en los 90 días posteriores al intervalo de fecha establecido en el análisis de los datos:

```
df_future = model_prophet.make_future_dataframe(periods=90)
df_pred = model_prophet.predict(df_future)
model_prophet.plot(df_pred)
```

Imágen Nº 10 - Serie de Tiempo libreria Prophet



Los puntos negros son las observaciones reales del precio de las acciones de YPF. La línea azul representa el ajuste que no coincide exactamente con las observaciones, ya que el modelo suaviza el ruido en los datos. Una característica importante es que Prophet cuantifica la incertidumbre, que está representada por los intervalos azules alrededor de la línea ajustada. Lo que continúa hacia la derecha, después de los puntos negros, son los valores pronosticados en un lapso de 90 días. Prophet fue diseñado para analizar series de tiempo con observaciones diarias (lo que no significa que no haya formas de usar datos semanales o mensuales) que exhiben patrones en diferentes escalas de tiempo (semanal, mensual, anual, etc.).

Prueba de Dickey-Fuller aumentado (ADF)

La hipótesis nula de la prueba de Dickey-Fuller aumentado (ADF) establece que la serie temporal no es estacionaria. Con un p valor de 0.88 , no se tiene motivos para rechazar la hipótesis nula, lo que significa que se puede concluir que la serie no es estacionaria. El resumen de la prueba arrojó el siguiente resultado:

Test Statistic	**-0.557252**
p-value	**0.880352**
# of Lags Used	**2.000000**
# of Observations Used	**214.000000**
Critical Value (1%)	**-3.461282**
Critical Value (5%)	**-2.875143**
Critical Value (10%)	**-2.574020**
dtype: float64	

Prueba Kwiatkowski-Phillips-Schmidt-Shin (KPSS)

La hipótesis nula de la prueba Kwiatkowski-Phillips-Schmidt-Shin (KPSS) es que la serie temporal es estacionaria. Con un valor p de 0.01, tenemos razones para rechazar la hipótesis nula a favor de la alternativa, lo que significa que la serie no es estacionaria. El resumen de la prueba arrojó el siguiente resultado:

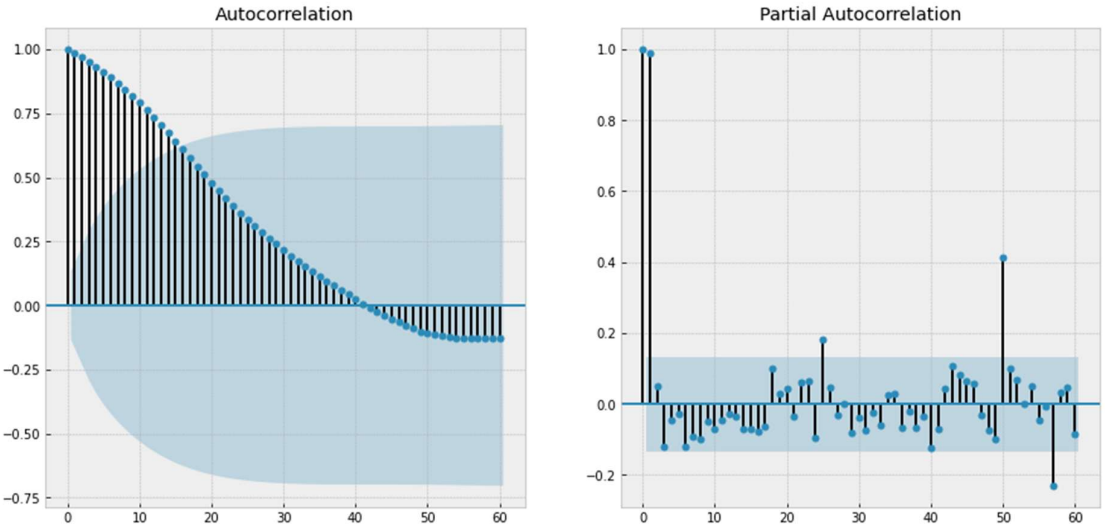
p-value is smaller than the indicated p-value	
Test Statistic	**0.940661**
p-value	**0.010000**
# of Lags	**15.000000**
Critical Value (10%)	**0.347000**
Critical Value (5%)	**0.463000**
Critical Value (2.5%)	**0.574000**
Critical Value (1%)	**0.739000**
dtype: float64	

Gráficos de la función de autocorrelación (parcial) (PACF / ACF)

En el gráfico de la función de autocorrelación (parcial), podemos ver que hay autocorrelaciones significativas (por encima del intervalo de confianza del 95%, que corresponde al nivel de significancia seleccionado del 5%). También hay algunas autocorrelaciones significativas en los rezagos 1 y 4 en el gráfico PACF.

```
N_LAGS = 60
SIGNIFICANCE_LEVEL = 0.05
fig, ax = plt.subplots(1, 2, figsize=(15, 7))
plot_acf(CloseAdj.y, ax=ax[0], lags=N_LAGS, alpha=SIGNIFICANCE_LEVEL)
plot_pacf(CloseAdj.y, ax=ax[1], lags=N_LAGS, alpha=SIGNIFICANCE_LEVEL)
```

Imágen Nº 11 - Autocorrelación y Autocorrelación parcial



Modelos ARIMA

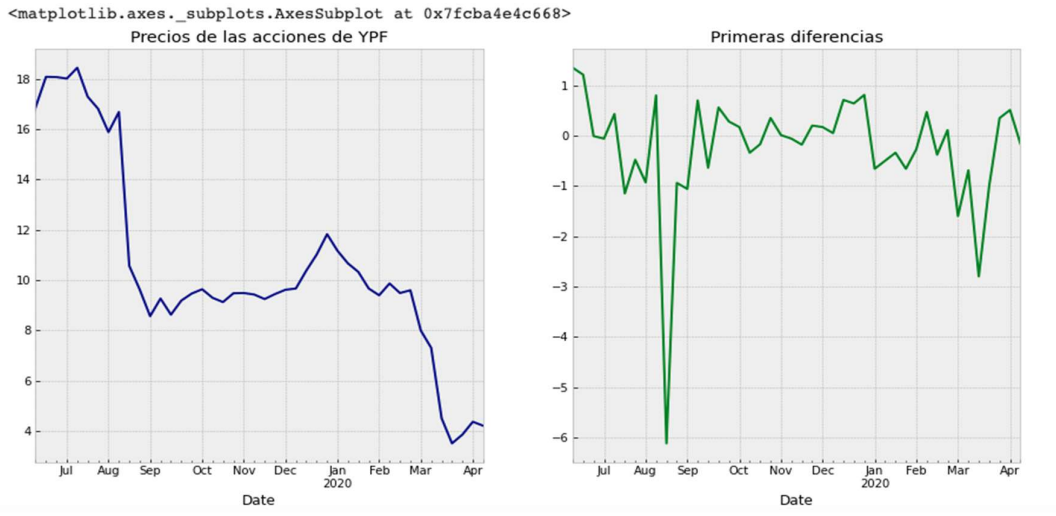
Los modelos ARIMA son una clase de modelos estadísticos que se utilizan para analizar y pronosticar datos de series temporales. Su objetivo es hacerlo describiendo las autocorrelaciones en los datos. ARIMA significa Autoregressive Integrated Moving Average y es una extensión de un modelo ARMA más simple. El objetivo del componente de integración adicional es garantizar la estacionariedad de la serie, porque, en contraste con los modelos de suavizado exponencial, la clase ARIMA requiere que las series temporales sean estacionarias (Troiano et al., 2020).

Para ello, primero se cargará las librerías necesarias para la ejecución de los modelos, luego se desarrollará el algoritmo específico. La sintaxis es la siguiente:

```
from statsmodels.tsa.arima_model import ARIMA
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.stats.diagnostic import acorr_ljungbox
import scipy.stats as scs
```

En el siguiente gráfico, podemos ver una tendencia más o menos lineal en el precio de las acciones de YPF, que indica no estacionariedad de la serie temporal.

Imágen Nº 12 - Modelo ARIMA



La sintaxis para generar los gráficos es la siguiente:

```
df_diff = df.diff().dropna()
plot.style.use('bmh')
fig, ax = plt.subplots(1,2, figsize=(15, 7), sharex=True)
df.plot(title = "Precios de las acciones de YPF", ax=ax[0], color=['darkblue'])
df_diff.plot(ax=ax[1], title='Primeras diferencias', color=['green'])
```

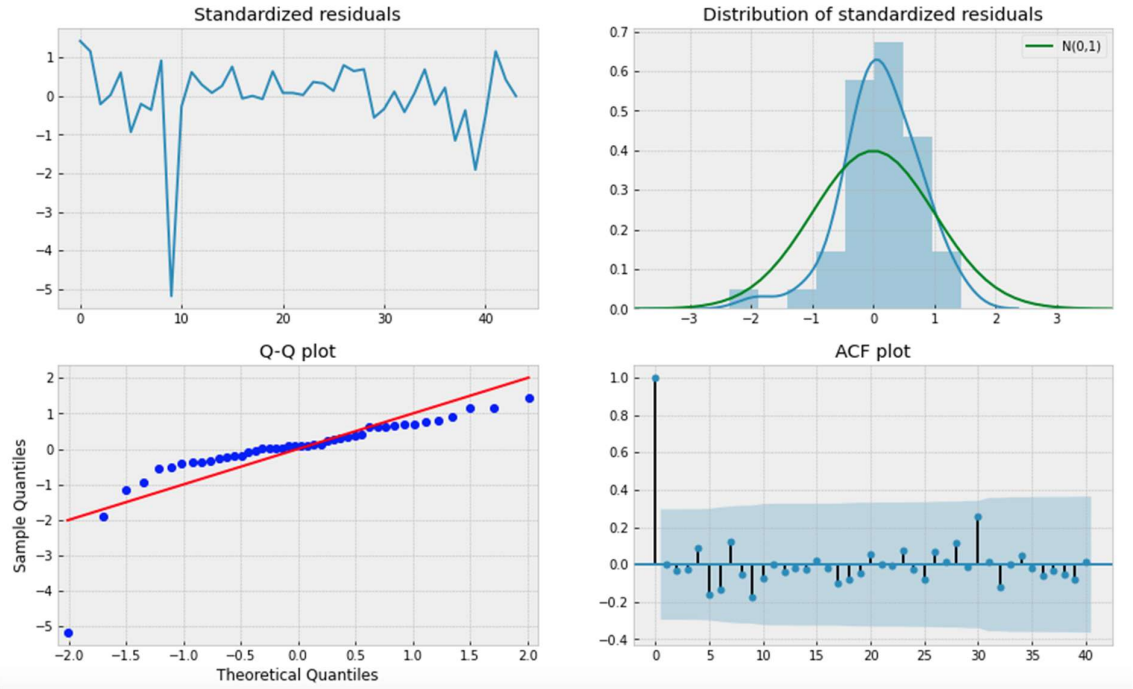
A continuación se desarrolla un algoritmo para ajustar el modelo ARIMA en función de los residuos:

```
def arima_diagnostics(resids, n_lags=40):
    # create placeholder subplots
    fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(15, 7))
r = resids
    resids = (r - np.nanmean(r)) / np.nanstd(r)
    resids_nonmissing = resids[~(np.isnan(resids))]
…
…
…

    return fig
```

Para visualizar el código completo, dirigirse al repositorio de GitHub.

Imágen Nº 13 - Modelo ARIMA en función de los residuos

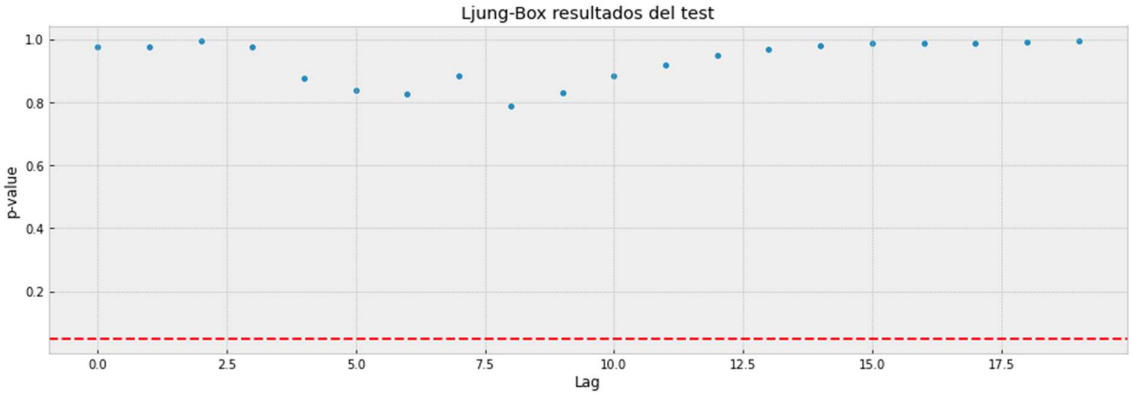


La distribución de los residuos estandarizados se parece a la distribución normal. El promedio de los residuos es cercano a cero llegando a la conclusión de que esos residuos no están correlacionados. Estas características hablan a favor de un buen ajuste. Sin embargo, las colas de la distribución son ligeramente más pesadas que en condiciones normales, lo que podemos observar en el gráfico Q-Q plot.

Prueba de Ljung-Box

Los resultados de la prueba de Ljung-Box no nos dan ninguna razón para rechazar la hipótesis nula de no autocorrelación significativa para ninguno de los rezagos seleccionados. Esto indica un buen ajuste del modelo.

Imágen Nº 14 - Modelo Ljung-Bo



Simulaciones de Monte Carlo

Las simulaciones de Monte Carlo son algoritmos que utilizan muestreo aleatorio repetido para resolver problemas que tenga una interpretación probabilística. En finanzas, una de las razones por las que ganaron popularidad es que pueden usarse para estimar con precisión las integrales.

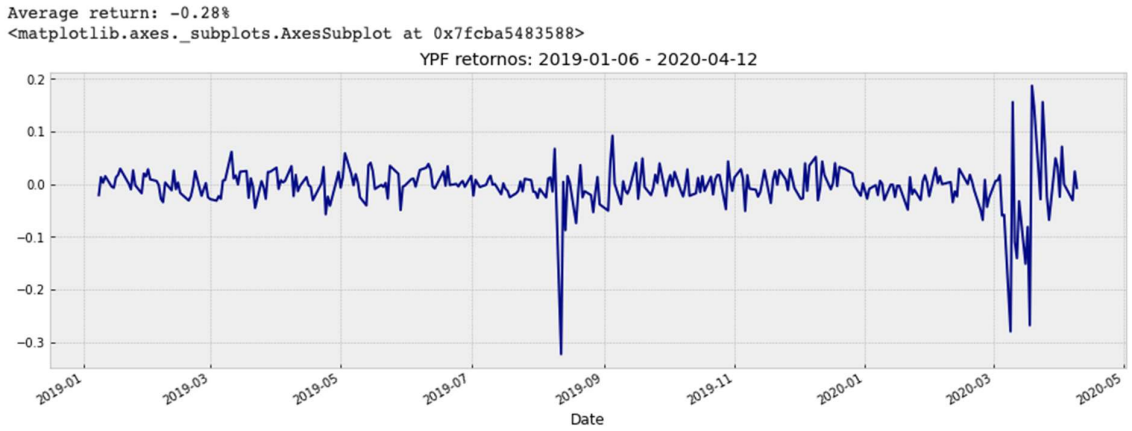
La idea principal de las simulaciones de Monte Carlo es producir una multitud de rutas de muestra: posibles escenarios y resultados, a menudo durante un período de tiempo determinado. El horizonte se divide en un número específico de pasos de tiempo y el proceso de hacerlo se llama discretización.

Su objetivo es aproximar el tiempo continuo, ya que el precio de las acciones de YPF (activos bajo análisis) ocurre en tiempo continuo. Los resultados de todas estas rutas de muestra simuladas se pueden usar para calcular métricas como el porcentaje de veces que ocurrió un evento, el valor promedio de un instrumento en el último paso, y así sucesivamente. Históricamente, el principal problema con el enfoque de Monte Carlo era que requería una gran potencia computacional para calcular todos los escenarios posibles. Hoy en día, se está volviendo menos problemático ya que se puede ejecutar las simulaciones bastante avanzadas en cualquier dispositivo.

El código para calcular los retornos diarios es el siguiente:

```
adj_close = df['Adj Close']
returns = adj_close.pct_change().dropna()
print(f' Average return: {100 * returns.mean():.2f}%')
returns.plot(title=f'{RISKY_ASSET} retornos: {START_DATE} - {END_DATE}', figsize=[16, 5], color=['darkblue'])
```

Imágen Nº 15 - Simulación Monte Carlo - Retornos diarios



El promedio de retorno diario para el periodo bajo análisis es del -0.28%

Ahora se puede dividir el conjunto de datos bajo análisis, en dos subconjuntos, uno de entrenamiento y otro de prueba, con el objetivo de realizar las simulaciones, en el presente ejemplo, se efectuarán 100 simulaciones simultáneas.

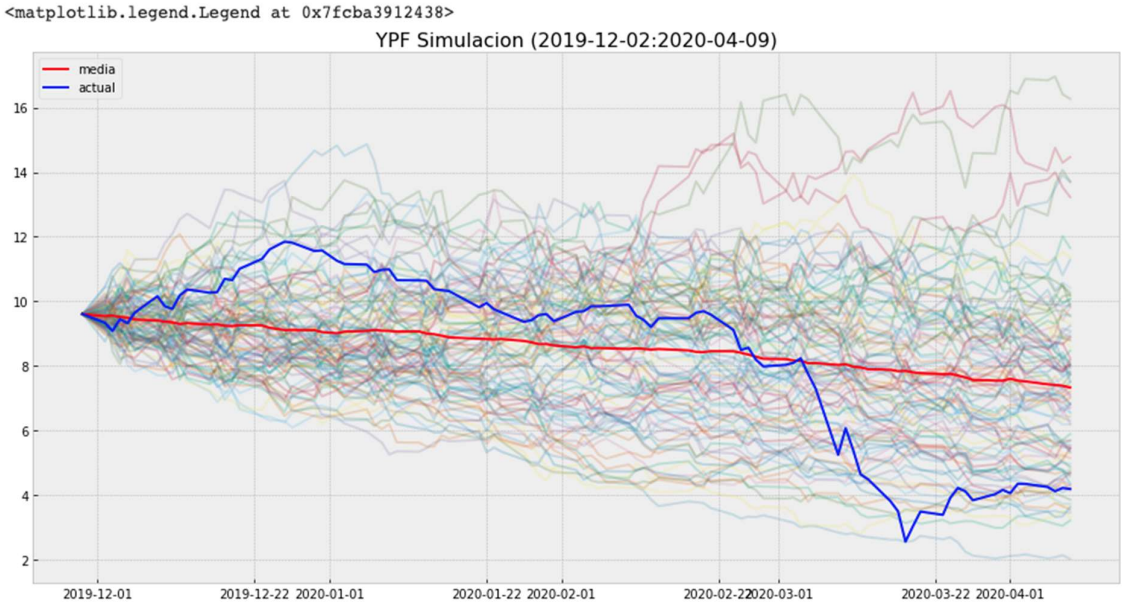
```
train = returns['2019-06-01':'2019-11-30']
test = returns['2019-12-01':'2020-04-12']
```

De esta forma, el periodo se ha partido en dos. Luego se define el número de simulaciones, y se crea el algoritmo para la simulación:

```
def simulate_gbm(s_0, mu, sigma, n_sims, T, N):
  dt = T/N
  dW = np.random.normal(scale = np.sqrt(dt), size=(n_sims, N))
  W = np.cumsum(dW, axis=1)
  time_step = np.linspace(dt, T, N)
  time_steps = np.broadcast_to(time_step, (n_sims, N))
  S_t = s_0 * np.exp((mu - 0.5 * sigma ** 2) * time_steps + sigma * W)
  S_t = np.insert(S_t, 0, s_0, axis=1)
  return S_t
```

Luego se ejecuta la simulación, y se trazan todos los resultados:

Imágen Nº 16 - Simulación Monte Carlo



Hay que tener en cuenta que la visualización sólo es factible para un número razonable de rutas de muestra.

En casos de la vida real, queremos utilizar significativamente más rutas de muestra que 100, ya que generalmente, a mayor número de rutas de muestra, más precisos y confiables son los resultados que se obtienen.

Se puede observar del gráfico que la media tiene un comportamiento descendente.

Resultados obtenidos

Con el desarrollo de este punto, se deja constancia que se alcanzan los objetivos de los ejes centrales de la propuesta de investigación:

- Esbozar modelos de aprendizaje profundo a datos de activos pertenecientes a los ecosistemas de la economía digital, puntualmente a las aplicaciones verticales asociadas a la actividad financiera.
- Aplicar modelos generados con aprendizaje profundo, con el fin del control de datos provenientes de la economía digital.
- Estudiar algoritmos y aplicaciones utilizables.

Aprendizaje Profundo

En los últimos años, se han visto muchos éxitos logrados mediante la aplicación de algoritmos de aprendizaje profundo. Las redes neuronales, en todas sus variantes se aplicaron con éxito a tareas en las que los algoritmos tradicionales de aprendizaje automático no podían tener éxito. La aplicación que se desarrollará tiene que ver con las predicciones de series de tiempo, y puede adaptarse tanto para series univariadas como multivariadas.

Como hipótesis se sabe que las series de tiempo financieras son erráticas y complejas, de ahí la razón por la cual es tan difícil modelarlas. Los enfoques de aprendizaje profundo son especialmente aptos para esa tarea.Como primer medida, se habilitaran las librerías necesarias para poder desarrollar los modelos de aprendizaje profundo, de acuerdo a la siguiente sintaxis:

```
import yfinance as yf
import numpy as np
import tensorflow as tf
from tensorflow import keras
import torch
import torch.optim as optim
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import (Dataset, TensorDataset, DataLoader, Subset)
from sklearn.metrics import mean_squared_error
from sklearn.neural_network import MLPRegressor
device = 'cuda' if torch.cuda.is_available() else 'cpu'
```

Luego se definen los parámetros del conjunto de datos y de la red neuronal, de acuerdo a la siguiente sintaxis:

```
# data
TICKER = 'YPF'
START_DATE = '2019-01-06'
END_DATE = '2020-04-12'
N_LAGS = 3

# neural network
VALID_SIZE = 12
BATCH_SIZE = 5
N_EPOCHS = 200
```

La arquitectura de redes neuronales que se utiliza es la denominada perceptrones multicapa (MLP). Luego hay que definir un algoritmo para que se pueda transformar una serie temporal en un conjunto de datos apto para la red neuronal MLP, con la siguiente sintaxis:

```
def create_input_data(series, n_lags=1):

  X, y = [], []
  for step in range(len(series) - n_lags):
    end_step = step + n_lags
    X.append(series[step:end_step])
    y.append(series[end_step])
  return np.array(X), np.array(y)
```

Luego hay que transformar las series de tiempo consideradas. en entradas para la MLP.

```
X, y = create_input_data(prices, N_LAGS)

X_tensor = torch.from_numpy(X).float()
y_tensor = torch.from_numpy(y).float().unsqueeze(dim=1)
```

En este paso ya se empiezan a definir los tensores, y se los puede verificar con el siguiente comando, para la interacción 0.

```
next(iter(train_loader))[0]
```

El resultado de la verificación es el siguiente, lo que en principio denota el funcionamiento de los tensores.

```
tensor([[16.2241, 14.0311, 13.9021]])
```

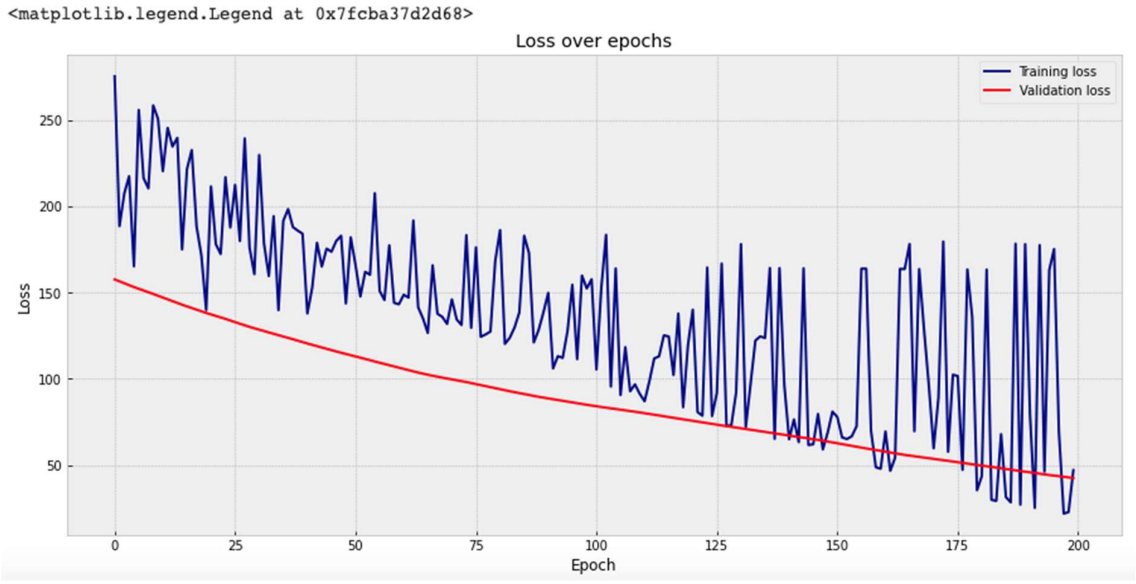
Como punto de partida se utilizará la denominada predicción ingenua (Naive Prediction) y se evalua el rendimiento. Luego se define la arquitectura de la red y se desarrolla el modelo de aprendizaje profundo. Posteriormente se comprueba el modelo y se entrena la red neuronal, y la ejecución de ese código arroja por resultado:

```
<0> –    Train. loss: 275.51        Valid. loss: 157.70
<50> –   Train. loss: 165.98        Valid. loss: 113.05
<100> –  Train. loss: 105.51        Valid. loss: 84.14
<150> –  Train. loss: 77.90                 Valid. loss: 62.69
Lowest loss recorded in epoch: 199
```

Luego se grafican las pérdidas y las validaciones, previamente se desarrolla el siguiente algoritmo:

```
train_losses = np.array(train_losses)
valid_losses = np.array(valid_losses)
fig, ax = plt.subplots(figsize=(15, 7))
ax.plot(train_losses, color='darkblue', label='Training loss')
ax.plot(valid_losses, color='red', label='Validation loss')
ax.set(title="Loss over epochs",
    xlabel='Epoch',
    ylabel='Loss')
ax.legend()
```

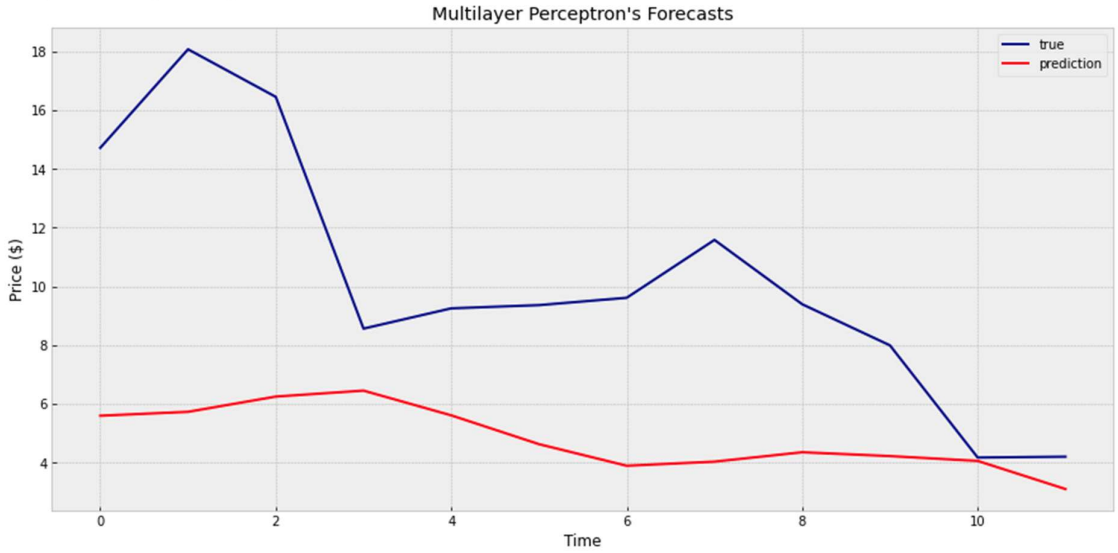
Imágen Nº 17 - Diagrama aprendizaje profundo



Ahora se carga el mejor modelo, que es el que tiene la perdida de validación más baja, y luego se obtienen las predicciones.

Imagen Nº 18 - Diagrama aprendizaje profundo - Predicción

```
MLP's forecast — MSE: 42.53, RMSE: 6.52
<matplotlib.legend.Legend at 0x7fcba36cc828>
```



Usando el perceptrón multicapa, se obtienen resultados mejores que con el pronóstico ingenuo.

Conclusiones y Proyecciones

Las conclusiones desarrolladas serán plasmadas en virtud de los tres objetivos establecidos en los ejes centrales de la propuesta de investigación.

Conclusiones

La digitalización de la economía, como consecuencia natural del avance tecnológico de las tecnologías de la información y de la comunicación, facilitando con ello, el desarrollo de la inteligencia de negocios, es decir la inteligencia artificial y los procesos inherentes de esa revolución, impactan en los procesos de negocios del ecosistema digital de diferente forma.

En la presente investigación se desarrolla un ejemplo de obtención de datos, materia prima del proceso de extracción del conocimiento, y se transitan las etapas clásicas como limpieza y transformación de los datos, visualizaciones, para llegar a la aplicación de algoritmos y generación de un modelo de aprendizaje profundo con redes neuronales para análisis predictivo de los datos. Esos datos bajo análisis son datos financieros, son datos de un activo, y todo ha sido posible con un dispositivo con acceso a internet.

Ahora bien, la simplicidad del proyecto se vio acotada al análisis de un solo tipo de acciones. Ese análisis y las conclusiones obtenidas del desarrollo de los modelos de aprendizaje profundo, facilitan e incentivan las futuras transacciones comerciales, por contar con información, que puede convertirse en conocimiento. Es decir, la analítica de datos impacta en los procesos de negocios de la economía digital, e impactará más aún cuando al análisis, a los algoritmos y a los modelos se les suman múltiples variables, hoy al alcance de todos, aún no aprovechadas por las personas que conforman el ecosistema digital.

Ni decir las ventajas competitivas de este conocimiento para los profesionales de ciencias económicas, vislumbrando competencias digitales en un muy próximo futuro. Conocer acerca de la inteligencia artificial, y de las herramientas disponibles, hoy en día tiene muchísimo valor.

Proyecciones

En virtud de los objetivos alcanzados en la presente investigación, se proyecta sumar otras variables al desarrollo de los modelos de aprendizaje profundo tales como:

- Incorporar análisis de procesamiento del lenguaje natural.

- Adicionar análisis de sentimiento y contrastarlo con valoración de los activos.
- Clasificar los distintos modelos de redes neuronales.

Referencias Bibliográficas

Digital Economy. Emerging Technologies and Business Innovation. (s. f.). Recuperado de https://www.academia.edu/33924983/Digital_Economy._Emerging_Technologies_and_Business_Innovation

Google Colaboratory. (s. f.). Recuperado 18 de enero de 2020, de https://colab.research.google.com/notebooks/welcome.ipynb

Negocios + Digitales. (s. f.). Recuperado de https://setconsulting.com.ar/landing/descarga/Negocios_+_Digitales_-_eBook.pdf

Pandas datareader documentation. (s. f.). Recuperado 12 de abril de 2020, de https://pandas-datareader.readthedocs.io/en/latest/

Parrilla, J. A. C. (2018). *Bienes digitales. Una necesidad europea.* Dykinson.

Políticas 4.0 para la cuarta revolución industrial. (2018, diciembre 3). Recuperado 11 de abril de 2020, de Puntos sobre la i website: https://blogs.iadb.org/innovacion/es/politicas-de-transformacion-digital/

Raquel, A. S., Ángel, S. A., & Rodrigo, M. G. (2019). *La transformación digital en el Sector Financiero*. Editorial UNED.

Souiden, I., Brahmi, Z., & Lafi, L. (2017). Data Stream Mining Based-Outlier Prediction for Cloud Computing. En R. Jallouli, O. R. Zaïane, M. A. B. Tobji, R. S. Tabbane, & A. Nijholt (Eds.), *Digital Economy. Emerging Technologies and Business Innovation—Second International Conference, ICDEc 2017, Sidi Bou Said, Tunisia, May 4-6, 2017, Proceedings* (pp. 131–142). doi: 10.1007/978-3-319-62737-3_11

Troiano, L., Bhandari, A., & Villa. (2020). *Hands-On Deep Learning for Finance*. Recuperado de https://subscription.packtpub.com/book/data/9781789613179