00 INTRODUCCIÓN

Este notebook contiene la **opción A del ejercicio 3**:

- Solo tienes que completar una de las dos opciones (A o B)
- Tienes 3 horas en total, y se aconseja que escojas la opción en los primeros 5-10 minutos, pues los ejercicios son largos
- Las distintas secciones del ejercicio incluyen una estimación del tiempo que se espera que te lleven, tenlo en cuenta para no emplear demasiado en uno en concreto.
- Si te atascas en alguna pregunta o no te resta mucho tiempo, deja indicado por escrito (usando una celda de tipo texto, por ejemplo) cómo la resolverías.
- Se valorarán los comentarios, orden y limpieza del código, no solo su funcionalidad.
- Las preguntas son en su gran mayoría independientes. En distintos puntos del ejercicio se te proporcionará el dataset completamente transformado (como se esperaría hasta ese momento) para que puedas seguir con los siguientes ejercicios hayas completado los anteriores o no. Por lo que, de nuevo, evita quedarte atascado en uno en concreto.

0001 Importa librerías

Puedes importar aquí las librarías habituales que creas necesitar, o hacerlo luego según las vayas necesitando.

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

01 Ingesta - 5 mins

Carguemos las fuentes de datos que utilizaremos en este ejercicio.

0101 - Datos Ibex

Primero, importemos datos diarios del Ibex 35. Están en el conocido formato OHLC:

- Precio de apertura ese día: **O**pen
- Precio máximo alcanzado ese día: **H**igh
- Precio mínimo alcanzado es día: **L**ow
- Precio de cierre ese día: **C**lose

010101 - Importa datos 1994 - 2020

Crea el dataframe **df_ibex_1** a partir de este .csv.

In [ ]:
```
url = "https://raw.githubusercontent.com/JotaBlanco/cnmv/main/data/IBEX-1994-2020.csv"
df_ibex_1 = pd.read_csv(url)
df_ibex_1.head()
```

010102 - Importa datos 2021

Ahora genera la variable **df_ibex_2**. Los datos están en el mismo formato, solo contiene información del año 2021.

In [ ]:
```
raw_url = "https://raw.githubusercontent.com/JotaBlanco/cnmv/main/data/IBEX-2021.csv"
df_ibex_2 = pd.read_csv(raw_url)
df_ibex_2.head()
```

010103 - Combina ambos

Combina ambos datasets en un unico dataframe llamado **df_ibex**. Además:

- Comprueba los nombres de las columnas en ambos casos para evitar desdobles (al juntarlos) de columnas que signifiquen lo mismo.
- El índice debe contener valores únicos en el dataset resultante (evita que queden varias filas con un mismo índice)

In [ ]:
```

```

In [ ]:
```
df_ibex
```

0102 - Tasa de paro y actividad

Este dataset contiene la evolución trimestrales de las tasas de paro y actividad, entre otros. Cárgalo en la variable **df_tasas**.

In [ ]:
```
url = "https://raw.githubusercontent.com/JotaBlanco/cnmv/main/data/detallado_evolucion_de_la_tasa_de_paro_en_espa%C3%B1a_desde_2002.csv"

df_tasas = pd.read_csv(url)
df_tasas.head(3)
```

02 Limpieza - 10 mins

Realicemos ahora ciertos cambios de formato para obtener un dataset final más útil y claro.

0201 Ibex

- Convierte la columna fecha a un formato temporal específico de pandas (datetime).
- Quédate solo con las columnas Date, Open, High, Low, AdjClose

In [ ]:

0202 Tasa de paro y actividad

- Primero, renombra las columnas:
  - 'Tasa de actividad (en %)' como 'Tasa Act'
  - 'Tasa de paro (en %)' como 'Tasa Paro'
- Convierte las columnas ['Activos', 'Ocupados', 'Parados', 'Tasa Act', 'Tasa Paro'] a formato float.

In [ ]:

In [ ]:

03 Transformaciones - 30 mins

Vamos a transformar ambas fuentes de información para obtener una granuladilad mensual.

In [ ]:
```
# Ejecuta esta celda para asegurar trabajar con la versión de datos esperad
a

# df_ibex
df_ibex = pd.read_csv("https://raw.githubusercontent.com/JotaBlanco/cnmv/ma
in/data/CLEAN_ibex_02.csv")
df_ibex["Date"] = [pd.to_datetime(date_i) for date_i in df_ibex["Date"]]

# df_tasas
df_tasas = pd.read_csv("https://raw.githubusercontent.com/JotaBlanco/cnmv/m
ain/data/CLEAN_tasas_02.csv")
```

0301 Ibex - 10 mins

En el dataset del ibex partimos de una granularidad diaria, por lo que deberemos agrupar dicha información a nivel mensual.

030101 Año y mes

Genera las columnas "año" y "mes" a partir de la fecha.

In [ ]:

030102 ROC

El término ROC (https://www.investopedia.com/terms/p/pricerateofchange.asp) (Rate of Change) se utiliza para describir el crecimiento relativo de un valor en un periodo.

Añade una nueva columna ROC que muestre la variación de precio al cierre diaria tal que: precio_cierre_hoy/precio_cierre_ayer

In [ ]:

030103 Agrupa mensualmente

Agrupa la información del ibex mensualmente. Ten en cuenta el significado de cada variable a la hora de decidir el método de agrupación de cada una para que mantengan su significado tras la agrupación.

Guarda esta agrupación en el nuevo dataframe **df_ibex_mensual**.

In [ ]:

0302 Tasa de paro y actividad - 10 mins

En el caso de este dataset, la granularidad es trimestral. Para convertirla en mensual estimaremos los valores intermedios con una interpolación lineal.

030201 Año y mes

A partir de la información de la columna trimestre, añade las columnas año y mes (suponiendo que la información a nivel trimestral se obtiene en el último mes de cada trimestre). Es decir, asumimos que 3T2022 es año: 2022 y mes: 9 (último del tercer trimestre).

In [ ]:

030202 Variables útiles

Quédate solo con las variables ['año', 'mes', 'Activos', 'Ocupados', 'Parados', 'Tasa Act', 'Tasa Paro'], en ese orden.

In [ ]:

030203 Desagrega la información

Ahora, crea un dataframe **df_tasas_mensual** donde tengamos una fila por mes. Interpola linearmente la información que falte.

In [ ]:

In [ ]:

In [ ]:

030204 ROC Tasa de Paro

Añade ahora la variable "Tasa de Paro ROC" con el mismo concepto visto en el punto 030102, esta vez aplicado a la tasa de paro.

Hazlo tanto en df_tasas como en df_tasas_mensual.

In [ ]:

In [ ]:

0303 Combinación mensual - 10 mins

030301 Combina ambos datasets

Combina las tablas df_ibex_mensual y df_tasas_mensual para obtener **df_mensual**.

In [ ]:
```
# Ejecuta esta celda para asegurar trabajar con la versión de datos esperada

# df_ibex
df_ibex_mensual = pd.read_csv("https://raw.githubusercontent.com/JotaBlanco/cnmv/main/data/CLEAN_ibex_mensual_03.csv")

# df_tasas
df_tasas_mensual = pd.read_csv("https://raw.githubusercontent.com/JotaBlanco/cnmv/main/data/CLEAN_tasas_mensual_03.csv")
```

In [ ]:

030302 Añade la columna fecha

Usando la información de año y mes, genera una nueva columna "Fecha" en formato datetime en la primera posición del dataframe.

In [ ]:

0304 Comodín

Antes de continuar, importa de nuevo las fuentes de información ya tratadas para asegurar que realizas las siguientes secciones con los datos procesados de la manera pretendida.

In [ ]:
```
df_ibex = pd.read_csv("https://raw.githubusercontent.com/JotaBlanco/cnmv/main/data/CLEAN_ibex.csv")
df_ibex["Date"] = [pd.to_datetime(date_i) for date_i in df_ibex["Date"]]
df_ibex.head()
```

In [ ]:
```
df_tasas = pd.read_csv("https://raw.githubusercontent.com/JotaBlanco/cnmv/main/data/CLEAN_tasas.csv")
df_tasas
```

In [ ]:
```
df_mensual = pd.read_csv("https://raw.githubusercontent.com/JotaBlanco/cnmv/main/data/CLEAN_mensual.csv")
df_mensual["Fecha"] = [pd.to_datetime(date_i) for date_i in df_mensual["Fecha"]]
df_mensual
```

04 Análisis - 60 mins

Estudiemos ahora estos datasets.

0401 Distribución de la variable ROC - 5 mins

Para la variable ROC de df_ibex, extrae los estadísticos básicos (media, cuantiles, mediana, etc.) y comenta la distribución brevemente.

In [ ]:

Tus comentarios (una o dos frases es suficiente):

--

--

0402 Evolución temporal - 5 mins

Muestra en una gráfica la evolución del valor del Ibex a lo largo del tiempo (utiliza el valor de cierre). Comenta la gráfica brevemente.

In [ ]:

In [ ]:

Comenta la evolución temporal brevemente (una o dos frases es suficiente):

—

—

0403 Comparación evolución temporal - 10 mins

Ahora, utilizando df_mensual, muestra además de la evolución del ibex, la de la tasa del paro y de la tasa de actividad. Todas en el mismo gráfico, usando distintos ejes Y como corresponda. Comenta brevemente el gráfico.

In [ ]:

Comenta la evolución temporal brevemente (una o dos frases es suficiente):

—

—

0404 Diagrama de dispersión - 10 mins

Muestra la asociación entre los dos ROCs (de la evolución del Ibex y de la tasa de paro):

- de manera visual con un diagrama de dispersión
- de manera cuantitativa calculando el coeficiente de correlación de pearson

Comenta los resultados.

In [ ]:

In [ ]:

Comenta la asociación entre variables brevemente (una o dos frases es suficiente):

__

__

0405 Responde a las siguientes preguntas - 15 mins

040501 ¿Qué día se dio la mayor subida del Ibex porcentual (ROC)?

In [ ]:

040502 ¿Cuál fue el día, precio de apertura y cierre el día que se dio la mayor bajada del Ibex porcentual?

In [ ]:

040503 ¿Cuántos días ha habido entre el 2015 y el 2020 (ambos incluídos) con una subida porcentual >5% al cierre respecto al cierre anterior?

In [ ]:

In [ ]:

040504 ¿En qué año se dieron más días donde la diferencia entre High y Low fue al menos 3 veces aquella entre la apertura y el cierre?

In [ ]:

0406 Crea una función - 15 mins

Crea la función "histograma_avanzado" que muestra el histograma, los cuantiles y la media en una única gráfica. La función requiere por parámetros:

- vector_x: sobre el que se realizará el histograma
- vector_intervalos: con los valores de los intervalos que se usarán para el intervalo

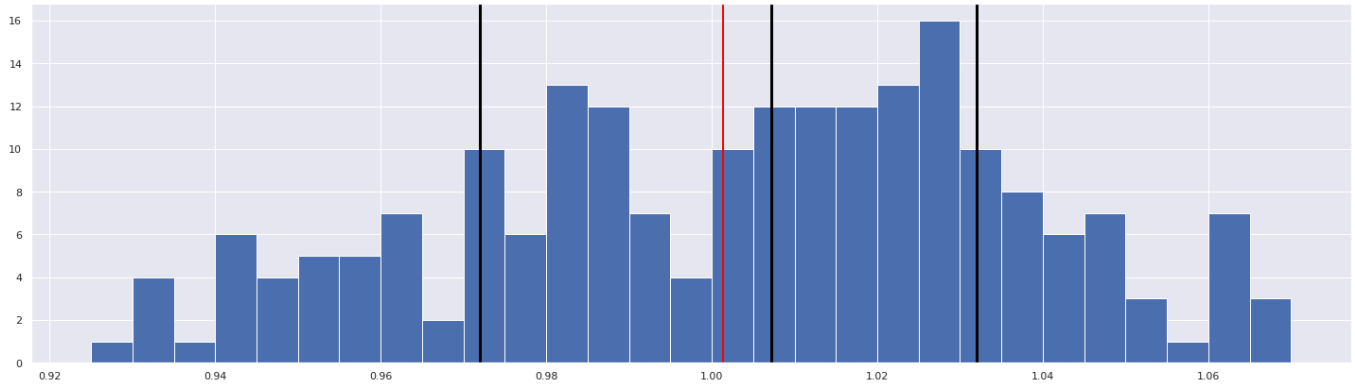
El diagrama producido contendrá:

- el histograma usando los intervalos seleccionados
- los cuantiles como líneas verticales en negro
- la media como líneas verticales rojas

Utiliza la función para estudiar y comentar las variables ROC, Ocupados y Tasa Paro.

In [ ]:
```
# Es decir, esto:
histograma_avanzado(df_mensual["ROC"], np.arange(0.925, 1.075, 0.005))
```

Devolverá esto:



In [ ]:
```
def histograma_avanzado(vector_x, vector_intervalos):
  """
  Define la función
  """
```

In [ ]:
```
histograma_avanzado(df_mensual["ROC"], np.arange(0.925, 1.075, 0.005))
```

In [ ]:
```
histograma_avanzado(df_mensual["Ocupados"], np.arange(df_mensual["Ocupado
s"].min(), df_mensual["Ocupados"].max(), 250))
```

In [ ]:
```
histograma_avanzado(df_mensual["Tasa Paro"], np.arange(0, 40, 2.5))
```

Tus comentarios (una frase por histograma es suficiente):

--

--

5 Estudio de fuente externa - 30 mins

El estudio de fuentes complementarias puede ser muy beneficioso. En este caso, vamos a analizar los comentarios sobre el ibex 35 en un foro público de inversión.

IMPORTANTE

*A lo largo de este ejercicio (5) encontrarás ayuda en forma de código y comentarios para completarlo empleando las librerías **BeautifulSoup y requests**. Sin embargo, tienes total libertad para realizarlo con otras librerías, o en general, obviando el código propuesto si consideras que no te resulta útil.*

5.1 Web Scrapping - 10 min

Vamos a utilizar BeautifulSoup para extraer todos los comentarios de la url https://es.investing.com/indices/spain-35-commentary/1 (https://es.investing.com/indices/spain-35-commentary/1) (no solo los visibles en la primera página del chat al que nos lleva la url) y formar con dicha información un dataframe con el nombre, apellido, mensaje, likes y no likes de cada mensaje.

Esta primera parte está solucionada para ti, ejecuta las celdas prestando atención y familiarizándote con el contenido de la sopa de HTML.

In [ ]:
```
import requests
from bs4 import BeautifulSoup
import json
```

In [ ]:
```
# Send a request to the chat's URL
url = "https://es.investing.com/indices/spain-35-commentary/1"

html = requests.get(url).content
soup = BeautifulSoup(html, 'html.parser')
print(soup.prettify())
```

In [ ]:
```
# Encuentra el script de tipo json y guárdalo
tag = soup('script', {'type': 'application/json'})[0]

# Carga la información en un diccionario
comments_dic = json.loads(tag.text)

# Inspecciona las llaves disponibles
display(comments_dic.keys())

# Con una inspección visual del contenido en cada llave se localiza la info
rmación de los mensajes
display(comments_dic['props'].keys())
display(comments_dic['props']['pageProps'].keys())
display(json.loads(comments_dic['props']['pageProps']["state"]).keys())
display(json.loads(comments_dic['props']['pageProps']["state"])["dataStor
e"].keys())

# Hasta llegar a la información de los mensajes
print()
print()
print("Mensajes:")
page = json.loads(comments_dic['props']['pageProps']["state"])["dataStore"]
["forumStore"]
page
```

5.2 Función message_to_df - 10 min

Crea la función message_to_df que convierta **un solo** mensaje del chat (en formato diccionario) a una fila de un dataframe de pandas con las columnas: ['id', 'userFirstName', 'userLastName', 'text', 'likes', 'dislikes', 'replies'] donde 'replies' hace referencia al número de mensajes en respuesta.

Importante: hay mensajes que continen otros que son respuestas en un mismo hilo. Cada respuesta será tomada como un nuevo mensaje y por tanto la función debe incluirlo como otra fila en el dataframe que devuelve.

In [ ]:
```
# Esto contiene una página de mensajes (incluyendo alguna información meta
como el número de página)
page
```

In [ ]:
```
# Esto únicamente los mensajes en esa página
messages_in_page = json.loads(page)["comments"]["_collection"]
messages_in_page
```

In [ ]:
```
# Por tanto, este es el primer mensaje. Un mensaje puede tener otros respon
diéndole en un mismo hilo
message_i = messages_in_page[0]
message_i
```

In [ ]:
```
# Crea la función que convierta message_i en un dataframe
```

In [ ]:
```
def message_to_df(message_i):
    """
    Define la función
    """

    return df
```

In [ ]:
```
message_to_df(message_i)
```

5.3 Función page_to_df - 10 min

Ahora, utilizando las funciones del apartado anterior, crea otra page_to_df que convierta en dataframe todos los mensajes en una página.

In [ ]:
```
messages_in_page
```

In [ ]:
```
def page_to_df(page_i):
    """
    Define la función
    """
    return df
```

In [ ]:
```
page_to_df(messages_in_page)
```

6 Modelo predictivo - 30 mins

Usando df_mensual como dataset, crea un modelo predictivo simple que prediga la Tasa de Paro del mes que viene. Emplea el dataframe df_mensual, usando las variables, algoritmos y métodos que prefieras.

Si hay pasos que no te da tiempo a ejecutar, déjalos descritos.

In [ ]:
```
df_mensual = pd.read_csv("https://raw.githubusercontent.com/JotaBlanco/cnmv/main/data/CLEAN_mensual.csv")
df_mensual["Fecha"] = [pd.to_datetime(date_i) for date_i in df_mensual["Fecha"]]
df_mensual.head()
```

Out[ ]:

	Fecha	año	mes	Open	High	Low	AdjClose	ROC	Ac
0	2001-04-01	2001	4	9308.299805	9904.000000	8784.799805	9760.990234	1.048634	17853.8
1	2001-05-01	2001	5	9761.000000	9854.599609	9370.599609	9500.690430	0.973333	17892.9
2	2001-06-01	2001	6	9498.900391	9663.900391	8634.099609	8878.391602	0.934500	17932.1
3	2001-07-01	2001	7	8878.400391	9104.900391	8074.500000	8479.991211	0.955127	18009.4
4	2001-08-01	2001	8	8480.000000	8684.900391	8083.799805	8321.090820	0.981262	18086.7

In [ ]: